PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 | A2

(11) International Publication Number: **WO 98/20434**

(43) International Publication Date: 14 May 1998 (14.05.98)

(21) International Application Number: PCT/US97/19719

(22) International Filing Date: 30 October 1997 (30.10.97)

(30) Priority Data:

08/745,899	7 November 1996 (07.11.96)	US
08/762,289	9 December 1996 (09.12.96)	US

(71) Applicant: VAYU WEB, INC. [US/US]; Suite 206, 309 Mamaroneck Avenue, White Plains, NY 10605 (US).

(72) Inventor: LENZ, Frederick, P.; 183 Old Field Road, Old Field, NY 11733 (US).

(74) Agents: MCNELIS, John, T. et al.; Fenwick & West LLP, Suite 700, Two Palo Alto Square, Palo Alto, CA 94306 (US).

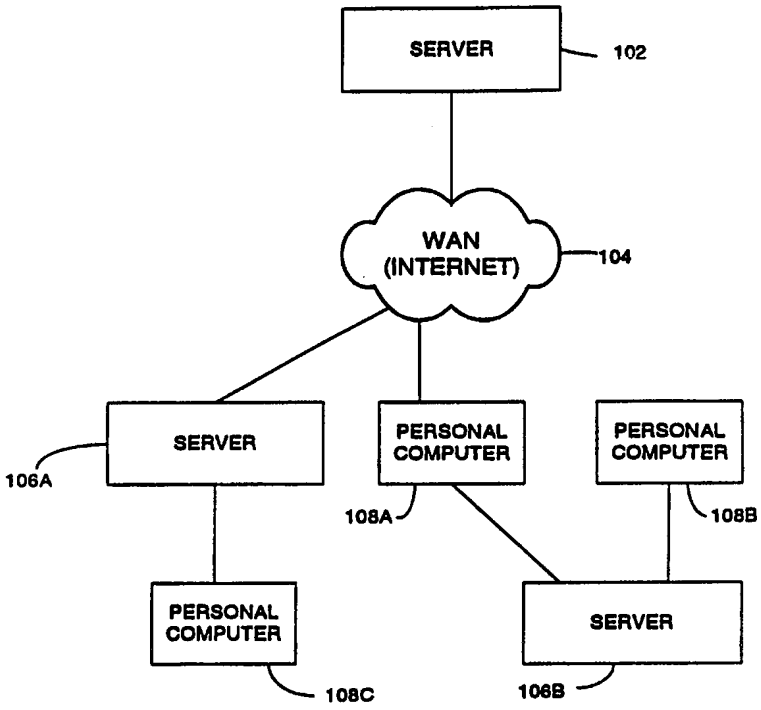
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: SYSTEM AND METHOD FOR DISPLAYING INFORMATION AND MONITORING COMMUNICATIONS OVER THE INTERNET

(57) Abstract

A system and method for creating, transmitting, and receiving web pages from a web server. The present invention enables a user to create a new web site by either creating new web pages or extracting information from previously created web pages. The invention enables a web site master to control the entire presentation of data to the user. In one embodiment of the present invention, a web site can include a first web page having a main menu. The present invention then permits the web site master to prepare a preset sequence of web pages that are displayed to the user for each menu item. The user of the web page selects one menu item from a menu display and then can passively receive information in a logical sequence. The control of the web page sequence can be used by advertisers, for example, to effectively present all information deemed helpful in attempting to convince the user to use the advertised product or service. In contrast to conventional web sites, a user accessing a web site developed according to the present invention will have a predefined sequence of web pages displayed in order to maximize the benefit to the web site master, e.g., to increase the behavioral modifications of the client due to controlled advertising. The present invention also increases the effectiveness of web sites having multiple web pages by using an artificial intelligence preloading methodology to reduce the delay in displaying a web page on the user's computer.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SYSTEM AND METHOD FOR DISPLAYING INFORMATION AND MONITORING COMMUNICATIONS OVER THE INTERNET

CROSS REFERENCE TO RELATED APPLICATION

This application is a continuation-in-part of U.S. Patent Application No. 08/745.899, filed by Frederick Lenz on November 7, 1996, entitled "System and Method for Displaying Information Over the Internet", Applicant's reference number 2608, which is incorporated by reference herein in its entirety.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to the field of Internet communication and more specifically to a system and method for generating and displaying world wide web site information.

2. Description of Background Art

The volume of traffic on and the number of users using the Internet and the world wide web (WWW) have increased significantly since the early 1990s and such increases are expected to continue. Previously, many users of the Internet and world wide web were technically sophisticated. Many, if not most, of the users in the future are expected to be technical novices. That is, many current and future users of the WWW will not understand "hyperlinks" and "applets." Many technically unsophisticated users of the Internet and WWW are intimidated by the current techniques required to move between pages in a web site. For example, one conventional technique for moving between pages in a web site involves searching through the text of a web page for one of many hyperlinks that may be of interest to the user and then selecting one hypertext or hypergraphic that is associated with the hyperlink to proceed to the page addressed by the hyperlink. While many technically sophisticated users do not have trouble understanding this web page addressing technique, less technically sophisticated users have trouble understanding hyperlinks and this web page navigation technique.

Another problem with conventional web sites is that the size of the data files that need to be downloaded by a user is increasing significantly. For example, it is not uncommon for a web site having complex images, e.g., color photographs, movie clips, or satellite images, to exceed several megabytes in size. A typical home user of a personal computer may have a maximum modem connection rate of 14.4 kilobits per second (kbps) which is approximately 1.8 kilobytes per second. Using such a modem, a web page whose data size is two megabytes requires over

18 minutes to download by the typical home computer, and only if the server that is sending the data is transmitting the data at the maximum permitted rate. While the data is being downloaded, conventional web sites display partial information on the screen of the users display monitor.

One growing use of the Internet and WWW is the creation of web sites for advertising. Advertisers are drawn by the low cost of the WWW. However, a significant drawback for advertisers is the slow rate of data transfer to typical home users. If an advertiser creates a web site where one page of the web site has a large size, e.g., two megabytes, then, as described above, a typical user having a 14.4 kbps modem will require over 18 minutes to download the single web page. A significant number of users will be reluctant to download such large advertisement web pages due to the large time commitment.

Another significant drawback for advertisers is that conventional web sites require the user to search hypertext and hypergraphics in each web page in order to proceed to what the user thinks is of interest. Therefore, an advertiser operating a web site having several pages frequently has users browse the web pages in a haphazard manner. However, successful advertisers prefer more control of the presentation sequence of an advertisement.

Accordingly, what is needed is a system and method for operating a web site that: (1) decreases the complexity of current web sites to reduce the intimidation of less sophisticated users of the web site; (2) decreases the waiting time between web page displays without increasing the data transmission rate; and (3) enables a web site owner to have more control over the presentation of web pages to a user.

SUMMARY OF THE INVENTION

The present invention is a new model for creating, transmitting, and receiving web pages from a web server. The present invention enables a user to create a new web site by either creating new web pages or extracting information from previously created web pages. The invention enables a web site master to control the entire presentation of data to the user. In one embodiment of the present invention, a web site can include a first web page having a main menu. The present invention then permits the web site master to prepare a preset sequence of web pages that are displayed to the user for each menu item. The user of the web page selects one menu item from a menu display and then can passively receive information in a logical sequence. The control of the web page sequence can be used by advertisers, for example, to effectively present all information deemed helpful in attempting to convince the user to use the

advertised product or service. In contrast to conventional web sites, a user accessing a web site developed according to the present invention will have a predefined sequence of web pages displayed in order to maximize the benefit to the web site master, e.g., to increase the behavioral modifications of the client due to controlled advertising.

The present invention also increases the effectiveness of web sites having multiple web pages by using an artificial intelligence preloading methodology to reduce the delay in displaying a web page on the user's computer.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is an illustration of a communication system in which the preferred of the present invention resides.

Figure 2 is a more detailed illustration of the server of the preferred embodiment of the present invention.

Figure 3 is a more detailed illustration of the server storage device of the preferred embodiment of the present invention.

Figure 4 is a flow chart describing the method performed by the server module according to one embodiment of the present invention.

Figure 5 is a flow chart describing the step of analyzing a web site according to one embodiment of the present invention.

Figure 6 is an illustration of a computer screen display showing information extracted from the web pages of the web site during the analyze web site 408 process according to one embodiment of the present invention.

Figure 7 is flow chart describing the step of configuring a modified web site according to one embodiment of the present invention.

Figure 8 is an illustration of a computer screen display for use in the web site configuration process according to one embodiment of the present invention.

Figure 9 is an illustration of an edit transition display according to one embodiment of the present invention.

Figure 10 is an illustration of a configuration image menu display according to one embodiment of the present invention.

Figure 11 is an illustration of a client computer storage device according to one embodiment of the present invention.

Figure 12 is a flow chart describing the method performed by the Vayu Web client module according to one embodiment of the present invention.

Figure 13 is a flow chart describing the method performed by the web page preload module according to one embodiment of the present invention.

Figure 14 is an illustration of a web page control panel according to one embodiment of the present invention.

Figure 15 is an illustration of a first menu web page according to an alternate embodiment of the present invention.

Figure 16 is an illustration of a second web site menu page according to an alternate embodiment of the present invention.

Figure 17 is an illustration of a site repository information according to an alternate embodiment of the present invention.

Figure 18 is an illustration of a screen display of the web site presentation properties of various web site sequences developed by the web site designer according to an alternate embodiment of the present invention.

Figure 19 is an illustration of a web page sequence properties screen display according to an alternate embodiment of the present invention.

Figure 20 is an illustration of a web page property screen display according to an alternate embodiment of the present invention.

Figure 21 is an illustration of a sequence optimizer screen display according to an alternate embodiment of the present invention.

Figure 22 is an illustration of the site layout screen display that is displayed to the web site designer according to an alternate embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A preferred embodiment of the present invention is now described with reference to the figures where like reference numbers indicate identical or functionally similar elements. Also in the figures, the left most digit(s) of each reference number correspond(s) to the figure in which the reference number is first used.

Figure 1 is an illustration of a communication system in which the preferred embodiment of the present invention resides. The communication system includes a web server 102, a wide area network 104, destination servers 106A, 106B, and personal computers 108A-C. The personal computers 108A-C are conventional personal computers, e.g., an Apple computer

or a Compaq computer. The personal computers 108 can be independent or part of a network, for example. The personal computers 108 can be connected directly to the WAN, using a modem, for example, or can be connected to the WAN 102 via a server 106A that uses a leased line, for example. The server 106A can operate using a conventional operating system, for example, UNIX or Windows NT. In order for a user of a computer, e.g., destination computer 108A to access data representing world wide web pages (web pages) from a web server 102, the user of the personal computer 108A identifies a uniform resource locator (URL) address to a web page in the web server 102 and transmits a request signal to the WAN 104. The WAN 104 is conventional, e.g., the Internet, and can use routers to route the request signal to the web server 102. A conventional web server receives a request and transmits the requested web page to the destination computer 108A. The destination computer 108A receives the web page data and displays the web page. This process can repeat several times for each web page requested by the user. The multiple requests per page are to obtain each multimedia object embedded in the page, e.g., pictures, sounds, and videos.

The present invention is a new model for creating, transmitting, and receiving web pages from a web server 102. The invention includes a server module and a client module. The server module is stored and implemented by the web server 102. The client module is stored in the web server 102 and is transmitted from the web server 102 to the destination computer 108A after the web server 102 receives a request from the destination computer 108A. The client module is executed by the destination computer 108A when displaying the web pages of the web site. A more detailed discussion of the server module and the client module is set forth below.

Figure 2 is a more detailed illustration of the web server 102 of the preferred embodiment of the present invention. As stated above, the web server is conventional and includes a conventional central processing unit 202, e.g., a Pentium Pro processor that is commercially available from Intel Corporation, Santa Clara, California, a conventional input/output system 204, conventional random access memory 206, and a conventional storage device 208 having unconventional computer application programs stored therein. The computer application programs stored in the storage device 208 are described in detail below.

Figure 3 is a more detailed illustration of the web server storage device 208 of the preferred embodiment of the present invention. The storage device 208 includes a server web site module 302, a web site project database 304 having menu information 308 and web page information 306, a web site analyzer 310, and a web site configuration module 312. The server

web site module 302 includes web site data including MIME files. MIME files can include MIME types that provide a technique for enabling web browsers, e.g., Netscape Navigator, to automatically launch a viewing program associated with a received file, hypertext markup language (HTML) data, and common gateway interface (CGI) data. The web site project databases 304 includes menu information 308 and data from web pages 306 extracted from the web page by the web site analyzer 310. A more detailed description of the web site project database is set forth below. The web site analyzer 310 analyzes each web page of the web site, extracts data from each web page, and stores the data in the web site project database 304. The web site configuration module 312 enables a user to create a menu and to create an ordered sequence of web pages that can be presented to a user (client) when a menu item is selected.

Figure 4 is a flow chart describing the method performed by the server module 102 according to one embodiment of the present invention. If the desired web pages of a web site do not yet exist 402 then the web pages are created 404 using conventional web page creation techniques. A web site creation description is provided in Chandler, Running a Perfect Web Site, Que Corporation (1995) that is incorporated by reference herein in its entirety. If the web pages that will be part of the web site exist 402 then the server 102 defines 406 a new site project, analyzes 408 the web pages of the web site, configures 410 the new web site, and saves 412 the web site. A more detailed description of steps 406, 408, 410, and 412 is set forth below. Alternatively, web pages for the web site can be developed concurrently using the present invention.

As stated above, the web server 102 defines 406 a new site project. In one embodiment a new site project is a data structure that is stored in the server storage module 208. When fully defined, the new site project includes or references all of the information about a web site. The information stored about each web site includes general information about the web site and a list of the web pages in the web site. The general information includes information such as: (a) the company's name where the company can be the party to which the web site information is directed, e.g., the company who is advertising on the web site; (b) the webmaster name, e.g., the creator of the web site; (c) the webmaster electronic-mail (e-mail) address. The list of web pages includes information about each web page, such as:(1) the URL of the web page; (2) a list of links out of the page; (3) a list of links into the page from other web pages in the web site; (4) the title of the web page; (5) the content type of the page, e.g., text, html, and image; (6) a list of pictures/videos/sounds in the page; and (7) the size of the web page. The size (in bytes) of the web page can be determined using any of a variety of techniques. In one embodiment, the size

of the web page is the sum of the number of bytes in the page's HTML, the size of all image <IMG>, sound <BGSOUND>, object <OBJECT>, applet <APPLET>, and script <SCRIPT> tag files. The information stored about each web page is determined during the steps of analyzing 408 the web site and configuring 410 the web site as described below.

Figure 5 is a flow chart describing the step of analyzing 408 a web site by the web site analyzer 310 according to one embodiment of the present invention. The web site analyzer determines the size of the web page and determines the links into and out of the page as described below. The web site analyzer 310 determines 502 if more web pages need to be analyzed. The web site analyzer 310 selects a web page by selecting one of the hyperlinks of a previously analyzed web page. The web page analyzer 310 analyzes the web page and sets a flag indicating that the web page has been analyzed. If the web site analyzer 310 determines that the web page has already been analyzed, then the web site analyzer sets a pointer to the memory location in the server storage 208 containing the information previously extracted during the web page analysis.

The web page analyzer 310 extracts 504 the title of the web page. In one embodiment, the title of the web page is extracted by identifying the text associated with the title tag (<TITLE>). The title can be displayed in order to identify the web page to the user. The web page analyzer 310 then extracts 506 the hyperlinks of the web page. In one embodiment, the hypertext links are extracted by identifying the anchor tags (<A>), the frame tags (<FRAME>), and map tags (<MAP>) in the web page. The web page analyzer 310 extracts 510 image file names by, for example, identifying the file associated with the image tag (<IMG>), and the web page analyzer 310 determines the size of the image file by (a) reading the HTTP size header or (b) downloading the file. In one embodiment, the HTTP size header is read since it is typically faster than downloading the file. The web page analyzer 310 also extracts 512 embedded file names in the web page by, for example, identifying the file associated with the embedded tag (<EMBED>). Then the web page analyzer 310 determines the size of each embedded file by (a) reading the HTTP size header or (b) downloading the file. The web page analyzer 310 also extracts 512 object files in the web page by, for example, identifying the file associated with the object tag (<OBJECT>). Then the web page analyzer 310 determines the size of each object file by (a) reading the HTTP size header or (b) downloading the file. The web page analyzer 310 also extracts 512 sound files in the web page by, for example, identifying the file associated with the bgsound tag (<BGSOUND>). Then the web page analyzer 310 determines the size of each sound file by (a) reading the HTTP size header; or (b) downloading the file. The web page

analyzer 310 extracts 514 class file names by, for example, identifying the file names associated with a class file name tag, e.g., <APPLET> in the Java programming language commercially available from Sun Microsystems, Inc., Santa Clara, California. The web page analyzer then determines a list of links out of the web page by downloading and scanning the contents of each class file. The web page analyzer 310 also extracts 516 additional information from the web page. For example, the web page analyzer 310 identifies scripts, e.g., Javascripts and VBScripts, associated with a script tag (<SCRIPT>), downloads the scripts and scans the scripts for links out of the web page. After extracting the above identified information, the web page analyzer 310 determines 518 the size of the web page by summing the number of bytes in (1) the HTML of the web page, and (2) all of the image/embed/class/bgsound/object/script tag files. Steps 504-518 are repeated for each web page.

Figure 6 is an illustration of a computer screen display showing information extracted from the web pages of the web site during the analyze web site process 408, described above, according to one embodiment of the present invention. The computer display illustrated in Figure 6 displays web page information including the web site name 602 "http://McJava/sampletree/morelink.htm", and the status of the analysis 612, a current page identifier 614, the total number of pages 616, the total number of links found in all of the analyzed pages 618. In addition, the display includes information concerning the parsed pages 620 and parsing details 622. The parsed pages 620 shows the list of web pages as they are being parsed. The pages are displayed in a tree structure format that shows the hierarchical nature of the web site. The parsing details 622 shows a detailed list of the links and files found in the page being currently parsed. The site name 602 is the URL of the web site being analyzed. Host information (not shown) can also be displayed. The host is the name of the host machine on which the web site resides. Similarly, a tree level indicator (not shown) can also be displayed. The tree level represents the hierarchical level of the web page in the web site. Any web page may exist at a number of levels (e.g., it may appear as a hyperlink in many different pages in the web site). The level represented here is the level closest to the top of the tree that the page appears. It will be apparent to persons skilled in the art that more or less information can be available to the webmaster when configuring the web site.

After the web site analyzer 310 analyzes 408 the web site, the web site configuration module 312 configures the web site. Although the web site analyzer 310 has searched and analyzed each web page in the web site, the web site analyzer 310 does not modify the original web site. The web site configuration module 312 enables the web site developer, e.g., a

webmaster, to construct a menu. In one embodiment of the present invention this menu is typically approximately 8-10 items, although it is envisioned that any number of items can be part of the menu. Each item will consist of a sequence of web pages that are displayed in a fixed order. In one embodiment of the present invention the web pages themselves are not modified except for the web page having the menu. In other embodiments the web pages are modified to, for example, ensure that all links out of the web page are deactivated, except for the links defined by the user. Figure 7 is a flow chart describing the step of configuring a modified web site by the web site configuration module 312 according to one embodiment of the present invention. The web site configuration module 312 enables the webmaster to create a menu structure that can be the first web page displayed in the web site. For example, Figure 8 is an illustration of a computer screen display for use in the web site configuration process according to one embodiment of the present invention. The display includes a menu configuration portion 802, a current web site structure portion 804, a status portion 814, and various control buttons that can be selected by the webmaster using a conventional mouse or other pointing device, for example. The control buttons in one embodiment include a menu shuffle up button 806, a menu shuffle down button 808, an add image button 810, an add transition button 812, and a status display 814.

The webmaster determines 704 if any more menu selections need to be modified. If the webmaster decides to modify or define a menu selection, the webmaster selects a menu, and defines 706 a sequence of web pages that will be displayed when the menu is selected. One process for assigning a web page to a menu is: (a) a user highlights the menu in the menu configuration portion 802 using the mouse or keyboard; (b) the user then highlights the web page in current web site structure portion 804; and (c) the user selects the Add button 816. The chosen web page will be added to the menu and will be displayed in the menu configuration portion 802 of the display. For example, a first menu is identified as "Luxury Vehicles" 820 in the menu configuration portion 802 of the display. The menu configuration portion 802 also includes information about the web pages that the webmaster has associated with the menu. For example, the top menu item is "Luxury Vehicles" 820. There are two web pages 822/824 associated with this menu item. These two web pages will be displayed in this order, i.e., web page 822 and then web page 824, when the user ultimately views this web site with their client PC and browser.

The shuffle up button 806 and the shuffle down button 808 enable a user to modify the web site structure by moving a web page up or down in the displayed menu configuration

portion 802 of the display. These buttons simplify the web site editing procedure for the web master. For example, a web page can be selected by the webmaster and associated with one of the menu items. The webmaster can then move, add, delete, or copy web pages in order to develop a suitable web site presentation sequence. The webmaster can also introduce transitions that can control the transition between web pages by selecting the "Add Transition" button 812. For example, a transition may be a timer of 10 seconds, or a keyclick from the user. In one embodiment the webmaster ensures that the user will view the web pages associated with each menu (if any) only in the predefined sequence by automatically showing the next page in response the transition event. For example, if a timer of 10 seconds has been selected by the web master then after 10 seconds has elapsed the next page will automatically be displayed. The user does not need to perform any action. In alternate embodiments, a limited number of page links are included that enable the user to advance to the next page, return to the previous page, or return to the main menu, for example.

As noted above, conventional web sites are created such that a client who accesses the web site can view the various pages in the web site in almost any sequence. One potentially significant use of the Internet and the WWW is in advertising. In one embodiment, the present invention can be utilized to create a web site having advertisement as a significant purpose of the web site. For example, an automobile company, e.g., General Motors, can develop a web site that advertises its currently available cars and trucks. One important aspect of advertising is the process of creating a need that can be satisfied by the advertiser. In order to efficiently and successfully accomplish this, it is helpful if the advertiser is able to control the sequence of information presented to the user. For example, if a car manufacturer developed a web site, they may not want the client to immediately jump to the web page having the cost of the vehicles. Instead, the advertiser could be more successful and effective if a description of the features and benefits of one or more car models were first shown to the user in order to familiarize the client with the cars and to therefore help persuade the client that the cars offered by the manufacturer will satisfy the needs of the client. After providing the information to the user, the web page having pricing information can be presented to the user. It will be apparent to persons skilled in the art that the above example is merely one example of how the current invention could be used and that many alternatives exist for developing and strategically planning the web page presentation sequence in a web site. The present invention provides the necessary tools to enable a webmaster to quickly, efficiently, and effectively generate a web site that gives control of the web site presentation to the web site owner instead of the client.

The page link can be displayed as a "Next Page" button or the webmaster can select to have one or more of the web pages transition automatically. Figure 9 is an illustration of an edit transition display 902 according to one embodiment of the present invention. In the example illustrated in Figure 9 the user has the ability to define the event that will trigger a web page transition for any particular web page, e.g., the "Leasing Information" web page as displayed in the menu item display area 904. The available types of transitions are displayed in the available transition portion 906 of the display. The webmaster selects one of the available transition types and activates the Add button 912, and the selection is displayed in the current transition portion 908 of the display. If the webmaster selects the timer option, the webmaster can set the delay between pages by entering the delay in the delay definition portion 910 of the display. To remove a selection from the Current Transition 908 display the web master uses the Remove button 914.

The webmaster also has the opportunity to provide 708 additional information concerning the web site and the individual web pages. For example, some of the information stored in the new site project data structure is defined by the webmaster, e.g., information concerning the company name, webmaster name, webmaster e-mail, and the URL of the main menu page or one or more web pages in the web site, for example. The webmaster has the opportunity to provide this information before or after the defining 706 the web page display sequence for the web site.

Figure 10 is an illustration of a configuration image menu dialog box display 1002 according to one embodiment of the present invention. The dialog box 1002 enables the user to generate a menu display by selecting a menu, e.g., Menu_4, and defining the active regions in the display portion 1006 for the selected menu. In Figure 10, the active region for Menu_4 ("New for '97") is defined in the active region definition dialog box 1008. In some embodiments of the present invention the webmaster does not create an initial menu page, instead only one web page sequence is defined and when the client selects the first page the sequence of the remaining web pages of the web site are displayed to the client in the predefined sequence.

After the webmaster defines the menus and defines the web page sequence associated with each menu, the modified web site is saved 412. In one embodiment of the present invention, the web site is saved as a series of files that are used when web site information is transferred to a client accessing the web site. In one embodiment these files are used to transfer the information gathered by the server tool to the client. The files include a stub Java Applet that is inserted into the home page of the site. One example of such a file is set forth in Table 1.

```
<APPLET CODE="browser.class" WIDTH="10" HEIGHT="10">

<PARAM NAME="menu_1" VALUE="[MENU_ITEM NAME='Luxury
Vehicles' AUTO=TRUE RECT=199,92,230,34 IMG=menu.gif]">

     <PARAM NAME="menu_1_1" VALUE="[LINK
     URL=http://www.xyzmotors.com/L2_N2.HTM SIZE=3514
     TRANSIT=TIMER DELAY=10]">

          <PARAM NAME="menu_1_1_1" VALUE="[LINK_OBJECT
          URL=http:// www.xyzmotors.com /Images\I_1_9.GIF
          SIZE=1059 CONTENT=image/gif]">

     <PARAM NAME="menu_1_2" VALUE="[LINK_FILE
     NAME=xyzmotors_menu_1.brw]">

<PARAM NAME="menu_2" VALUE="[MENU_ITEM NAME='Sport Utility'
RECT=199,129,229,39 IMG=menu.gif]">

     <PARAM NAME="menu_2_1" VALUE="[LINK URL=http://
     www.xyzmotors.com /L1_N1.HTM SIZE=12117
     TRANSIT=MENU]">

          <PARAM NAME="menu_2_1_1" VALUE="[LINK_OBJECT
          URL=http:// www.xyzmotors.com /Images\I_1_1.GIF
          SIZE=1257 CONTENT=image/gif]">

     <PARAM NAME="menu_2_2" VALUE="[LINK_FILE NAME=
     xyzmotors _menu_2.brw]">

<PARAM NAME="menu_3" VALUE="[MENU_ITEM NAME='Trucks and
Vans' RECT=198,173,230,47 IMG=menu.gif]">

     <PARAM NAME="menu_3_1" VALUE="[LINK URL=http://
     www.xyzmotors.com /L1_N1.HTM SIZE=12117 TRANSIT=TIMER
     DELAY=10]">

          <PARAM NAME="menu_3_1_1" VALUE="[LINK_OBJECT
          URL=http:// www.xyzmotors.com /Images\I_1_1.GIF
          SIZE=1257 CONTENT=image/gif]">

     <PARAM NAME="menu_3_2" VALUE="[LINK_FILE NAME=
     xyzmotors _menu_3.brw]">

<PARAM NAME="menu_4" VALUE="[MENU_ITEM NAME='Family Fare'
RECT=195,224,232,40 IMG=menu.gif]">

     <PARAM NAME="menu_4_1" VALUE="[LINK URL=http://
     www.xyzmotors.com /L1_N1.HTM SIZE=12117 TRANSIT=TIMER
     DELAY=10]">

          <PARAM NAME="menu_4_1_1" VALUE="[LINK_OBJECT
          URL=http:// www.xyzmotors.com /Images\I_1_1.GIF
          SIZE=1257 CONTENT=image/gif]">

     <PARAM NAME="menu_4_2" VALUE="[LINK_FILE NAME=
     xyzmotors _menu_4.brw]">
```

```
<PARAM NAME="menu_5" VALUE="[MENU_ITEM NAME='New For
'97']">

     <PARAM NAME="menu_5_1" VALUE="[LINK URL=http://
     www.xyzmotors.com /L1_N1.HTM SIZE=12117 TRANSIT=TIMER
     DELAY=10]">

          <PARAM NAME="menu_5_1_1" VALUE="[LINK_OBJECT
          URL=http:// www.xyzmotors.com /Images\I_1_1.GIF
          SIZE=1257 CONTENT=image/gif]">

     <PARAM NAME="menu_5_2" VALUE="[LINK_FILE NAME=
     xyzmotors _menu_5.brw]">

</APPLET>
```

Table 1

These parameters in Table 1 contain specific information about each link on the menu illustrated in the display portion 1006 of Figure 10, and the contents of the pages linked thereto. The menu items are passed as parameters. In the example set forth in Table 1, a naming convention is used to determine the order and structure of the menu. The data is structured with a format that is similar to HTML, but uses square brackets ([and]) instead of angle brackets (< and >). The first parameter (PARAM) is for the first item in the menu, named *menu_1*. Its type is *MENU_ITEM*, and the item's name is *Sport Utility*. The next *PARAM* is a page assigned to the *menu_1*, and its name is *menu_1_1*. The type is *LINK*, its *URL* is http://McJava/dealers/dealers.html, the *TRANSIT* is *TIMER*, and the *DELAY* is 10 seconds. The third *PARAM* is also assigned to the *menu_1* item, but it points to a file that contains the rest of the links for *menu_1*. The file name is *McJava_menu_2.brw*. There are one of these files for each menu item that has more than one link assigned to it. These files are used to limit the amount of information that is transmitted from the server to the client in the initial transmission. Since the number of web pages that could be assigned to all of the menu items could range into the hundreds and thousands, it is generally more efficient to send down a portion of menu information, and then retrieve more menu information as needed. The file structure design is extensible such that as new information is needed new name/value pairs are added. Older version of the software will ignore name/value pairs they do not understand. Newer version of the software will have reasonable defaults if they encounter an older file with missing name/value pairs.

As described above, a client utilizes a computer, e.g., a destination computer 108A, to access the defined web site stored on the server 102. The personal computer used by the client

can be a conventional personal computer, e.g., an IBM personal computer that is commercially available from IBM Corporation, Armonk, NY. The destination computer 108A includes a conventional CPU, conventional RAM, and a conventional storage device, that can be similar to the server 102 described above. Figure 11 is an illustration of a client computer storage device 1100 according to one embodiment of the present invention. The storage device 1100 includes a conventional Internet browser 1102, a Vayu Web client module 1104, a web page preload module 1106, menu files 1108, and a TCP/IP Internet connection protocol module 1110. The conventional Internet browser provides an Internet gateway to the destination computer 108A. The client uses the Internet browser 1102 to request access to the web site stored in the server storage module 208. The menu files 1108 include the files received from the server 102 in response to the access request. In addition, the personal computer storage module 1100 receives the Vayu Web client module 1104 and the web page preload module 1106 from the server 102. The operation of these modules is described in greater detail below.

In one embodiment, the Vayu Web client module has been implemented using three different techniques. These three techniques allow the client module to operate across the wide variety of client machines currently in use across the Internet, e.g., IBM PC, Apple Macintosh, Sun Microsystems Sparcstation, and across the wide variety of WWW Browsers currently in use, e.g., Netscape versions 2.n and 3.n, and Microsoft Internet Explorer versions 2.n and 3.n. The three techniques are: (1) as a Java Applet; (2) as a Javascript script where Javascript is a Internet language developed by Netscape Corporation; (3) as a Client-Pull mechanism using a <META> tag in the HTML file, that causes the browser itself to automatically load pages on a timed basis. The CONTENT argument contains the time in seconds. The URL contains the target URL. The <META> tag can be:

<META HTTP-EQUIV=REFRESH CONTENT="1 URL=target_url">

Figure 12 is a flow chart describing the method performed by the Vayu Web client module according to one embodiment of the present invention. As described above, the client requests information from the web site server 102 and the destination computer 108A downloads 1202 web site server information. The web site server information initially downloaded includes the Vayu Web client module 1104, the web page preload module 1106, and the new site project data structure that is stored in the menu files 1108. As described above, the new site project data includes information about the web site and includes references to web pages in the web site but does not include all of the web page data. The destination computer 108A implements 1204 a web page preload routine that is executed concurrently with the

remaining functions performed by the Vayu Web client module, as described below. Figure 13 is a flow chart describing the method performed by the web page preload module 1106 according to one embodiment of the present invention. The web page preload module 1106 gathers local environmental information concerning the client personal computer. This information includes the speed of the modem, the time of day, Internet service provider service level, browser type and version, and operating system type and version. This information is used by the artificial intelligence expert system to dynamically select a preload strategy to optimize the performance of the presentation of web pages to the client. The web page preload module 1106 preloads 1304 the first web page from each of the menus.

While the web page preload module 1106 is preloading the first web page associated with each of the menus, the Vayu Web client module 1104 displays 1206 the menu page and waits for the user to select a menu. When the user selects 1208 a menu item, the Vayu Web client module 1104 identifies 1210 the first web page in the predefined sequence of web pages associated with the menu item and displays 1212 the first web page. Before the user selects a menu item, the web page preload module 1106 preloads the first web page associated with each menu item, as stated above, and then determines the web page loading sequence for subsequent web pages based upon a determination of the most likely menu item selection. The web page preload module 1106 determines the most likely menu item selection based upon the selections of previous clients or, if no clients have previously accessed the web page, the webmaster may indicate which menu item to show first, or the first menu item is selected as the most likely to be selected. The web page preload module 1106 preloads web pages from the sequence of web pages associated with the most likely menu item selection by implementing the rule based expert system of the present invention, described below.

After the client selects a menu item, the Vayu Web client module 1104 and the web page preload module 1106 receive the web page sequence and web page information from the new site data structure in the menu files 1108. Specifically, the new site data structure includes the overall size of the web page, the number of embedded objects in each web page, and the size of the embedded objects in each web page, as described above. In one embodiment of the present invention, the web page preload module 1106 determines a priority for each web page in the web page sequence based upon the environmental parameters of the destination computer 108A, described above, a queue location of the web page in the predefined sequence, the size of the web page, the number of embedded objects in the web page, and the size of the embedded objects in the web page, for example. In addition, the web page preload module is keeping track

of how long it actually takes to load each page. From this information it calculates the actual throughput that the current client is getting. This throughput is used by the expert system to make decisions about preload strategies. For example, if the throughput falls below a certain value, the preload module may stop preloading certain types of MIME files (e.g., video, sound). The trigger value would be set by the web master during the menu configuration stage, described above. One example of a rule based expert system is set forth in Table 2.

```
; Set preload strategy based upon throughput
IF
     system.throughput < 6kpbs
THEN
     preload.order = STRATEGY_7
ENDIF

; Set preload strategy based upon throughput
IF
     system.throughput < 2kpbs
THEN
     mime.image = NOLOAD
ENDIF

; Set priority based upon size of an embedded object
IF
     object1.size > 250000
THEN
     object1.preload_priority = 5
ENDIF

; Set priority based upon size of an embedded object
IF
     object1.size > 500000
THEN
     object1.preload_priority = 7
ENDIF

; Set priority based upon size of an embedded object
IF
     object1.size > 1000000
THEN
     object1.preload_priority = 10
ENDIF

; Set priority based upon the number of embedded
; objects in a page
IF
     page1.objects > 5
THEN
     page1.preload_priority = 3
ENDIF

; Set priority based upon the number of embedded
; objects in a page
IF
```

```
     page1.objects > 10
THEN
     page1.preload_priority = 8
ENDIF

; Set preload order based upon browser type and time
of
; day
; MSIE_3x is Microsoft's Internet Explorer Version 3.x
IF
     ; night time hours
     browser.type = MSIE_3x AND (env.timeofday > 20:00
AND env.timeofday < 08:00)
THEN
     preload.order = SEQUENTIAL
ENDIF

; Set preload order based upon browser type and time
of
; day
; STRATEGY_7 is load page 1, then 2, then largest to
; smallest
IF
     ; business hours
     browser.type = MSIE_3x  AND (env.timeofday >
08:00 AND env.timeofday < 20:00)
THEN
     preload.order = STRATEGY_7
ENDIF
```

Table 2

After determining the preloading priority for each web page, the web page preload module 1106 preloads 1308 the web pages having the highest priority. The web page preload module 1106 monitors 1310 the preloading procedure to reduce any delay in displaying web page to the client. The monitoring function of the web page preload module 1106 permits the preload function to be dynamic in that if a problem occurs 1312 in the web page loading process, the web page preload module 1106 sorts and reorganizes 1314 the data. Examples of problems include: (a) when a page is needed to be displayed and has not yet been preloaded or (b) when the preloading is sufficiently ahead of the display so that the local PC web page cache is full, and pages must be discarded (and reloaded). When a problem occurs the expert system is re-initialized with the current data (including the throughput and modified menu files without the web pages that have already been displayed). A modified preload strategy is then determined 1306.

The client continues navigating 1210 through web pages while the destination computer 108A continues displaying 1212 web pages. As described above, in one embodiment of the present invention, the client can passively watch the sequence of web page displays. In alternate

embodiments, the user has the option of selecting a next page button. Within each web page the client also has the option of manipulating a control panel. Figure 14 is an illustration of a web page control panel 1400 according to one embodiment of the present invention. In this embodiment, the user has the option of returning to the beginning of the web page by selecting control panel button 1402, proceeding to the end of the web page (for example, a video clip embedded in the web page may be skipped) by selecting control panel button 1404, continuing or beginning the execution of the web page display by selecting control panel button 1406, pausing the execution of the web page by selecting control panel button 1408, decreasing the speed of web page execution by selecting control panel button 1410, increasing the speed of web page execution by selecting control panel button 1412, or stopping web page execution by selecting control panel button 1414.

An alternate (preferred) embodiment of the present invention is described below with reference to Figures 15-22. This alternate embodiment includes a standard structure and navigation for the menus (pages containing buttons) and menu items (buttons), allowing the user a set of options at each web page. One benefit of using the standard structure is to make the web site easier to use by the end user and to offer the web site designer more control over what is displayed to the user.

Figure 15 is an illustration of a first menu web page according to an alternate embodiment of the present invention. The first menu page 1500 illustrated in Figure 15 includes a single large button 1502 that includes the name of the company that is the subject of the advertising, e.g., "Vayu Web". The user can select the button 1502 to continue or the user can exit from the web site. In addition, the present invention enables the company to add a graphic display to the button or to the background to enhance the screen display.

The first menu web page can include a tool bar 1504 having various control buttons thereon. These control buttons are enabled and disabled for each web page based upon the web site definitions defined by the webmaster. When the user selects the button 1502, a sequence of pages may automatically be presented after which another menu page appears containing more items. Alternatively, the sequence can be skipped and the next menu page can be presented immediately. While the user is viewing a dynamic sequence, the control functions are typically disabled. The control functions include pause, fast forward, reverse, and exit. In alternate embodiments these control functions are enabled during the presentation of dynamic sequences. While a dynamic sequence is playing the user typically watches the display until the next menu page is reached.

Figure 16 is an illustration of a second web site menu page according to an alternate embodiment of the present invention. At the next web site menu page 1600, the user has three options, (1) return to the previous menu page, (2) select one of the menu items 1602, or (3) select the auto-play button 1604 to automatically cycle through the sequences following each menu item, for example. The order of cycling through the menu items is determined by the web site designer (web site master). Some items can be excluded from this "autoplay". A menu page can comprise any number of items. In this alternate embodiment the number of items is typically between 1 and 10. The web site designer can add their own graphics to the items or to the page background.

Certain web pages can be static, i.e., they do not automatically display another web page. Frequently, for static web pages, the "pause" and "forward" buttons on the tool bar 1504 are enabled. In this case, the page may include a description of a product and/or ordering information, for example. A maximum viewing time can be specified by the web site designer to proceed to the next page after some amount of time has elapsed, or the web site designer can permit the user to continue to the next web page by selecting a next page control button.

In this alternate embodiment, the tool bar 1504 is located at the top of the screen. The web site designer may choose not to display the tool bar 1504 for certain screens when, for example, the user has no other options within these screens but to wait for the next page. The tool bar 1504 can also include status indicators 1606 to show the remaining time left to view this page. This tool bar 1504 can be used for every page displayed using the present invention.

Another feature of the present invention is that the user is permitted to create multiple versions of each web site where the present invention permits optimization of each web site for different internet throughput rates. For example, the throughput range for different types of Internet access devices commonly used today varies from 14.4 kbps (modem) to 56 kbps (ISDN) to 1.5 megabits/sec (T1). This variation exceeds a factor of 100. Furthermore, users of the present invention on Intranets, e.g., local area networks (LANs) can reach data transfer rates exceeding 10 megabits/sec. Due to transfer rate constraints at the lower throughput speeds, the web site designer may include several versions of a web page, or several versions of a sequence, where a sequence includes a series of pages which are automatically displayed, with each version being a different size to optimize the web site display for different throughput environments. The database can include information about the various page and sequence versions. The artificial intelligence module of the present invention that was described above with respect to the expert system and is described in greater detail below, can automatically

select the appropriate page version depending on the throughput environment it has determined for a user. The Virtual Design Studio, which is the tool that the web site designer uses to assemble the web pages into advertisement sequences, is designed to assist the web designer in creating optimal sequences for various throughput environments. The Virtual Design Studio will also allow the web site designers to preview the web pages of the web site using a simulator of the present invention in order to simulate environments having various data throughput rates.

It will be apparent to persons skilled in the art that the use of different versions of the web pages is not required to operate the present invention. However, if the web site designer is expecting users with different throughput environments to access the web site, it may be advantageous to create versions of certain web pages that contain videos, sound, etc., so users with faster connections can view web pages having greater detail, while users with lower connections can still get the full benefits of this web site by viewing pages of a smaller size.

The present invention permits different versions of web pages and web sites by providing the web site designer with the option of providing high-quality (larger) pages for users connected at higher throughput rates and lower-quality (smaller) pages for users connected at lower throughput rates. The present invention enables the web site designer to easily mix the smaller pages with the larger pages when the throughput is fast. However, adding larger pages to a low-throughput connection requires the designer to carefully sequence the pages so a large page is preloaded while several smaller ones are being viewed. In addition, some pages take longer to view, for example if they contain detailed textual information that takes longer to read. This longer viewing time can be used to preload larger files. The Virtual Design Studio of the present invention includes modules which assist in the intelligent sequencing of pages in this manner not only to reduce the user's waiting time but also to improve the quality of the pages that can be viewed with slower connections.

Another factor to consider is that the actual throughput rate during an Internet connection session can vary greatly. For example, with a nominal connection speed of 28.8 kbps, the actual transfer rate can easily vary between 10 kbps and 28.8 kbps. Therefore, the artificial intelligence (AI) module of the present invention makes real-time adjustments during a connection session to keep the presentation flowing and prevent waiting time for the user. One of the techniques implemented by the AI module is to leave out certain pages that the designer has identified as being large but with low impact or information content. Another technique is to dynamically switch a higher-quality presentation to a lower one when the throughput slows

down, and then to switch back to the higher-quality one if the throughput picks up again. The artificial intelligence unit is described in greater detail below.

Figures 6, 8, 9, and 10 are illustrations of screen displays during the analyze web site process 408 and the configure web site process 410 for one embodiment of the present invention as described above. Figures 17-22 are illustrations of screen displays displayed to the web site designer during the analyze web site process 408 and the configure web site process 410 according to an alternate embodiment of the present invention. The Virtual Design Studio is stored in the web site analyzer 310 and enables a web site designer to: (1) construct presentation sequences from pre-existing and newly created web pages; (2) optimize the presentation sequences for different throughput environments and specify alternatives to be used by the artificial intelligence module to compensate for throughput variations in real time; and (3) create menu structures and user navigation through the web site. The virtual design studio includes a site repository information screen that is illustrated in Figure 17. The site repository information screen allows a web site designer to view, insert, delete and edit web page information and web page sequences. For example, the site repository information screen allows the web site designer to assemble pages into the sequences which form the basis of the automated web site content. The site repository information screen also enables the web site designer to modify and maintain the web site.

Figure 18 is an illustration of a screen display of the web site presentation properties of various web site sequences developed by the web site designer. Figure 18 illustrates two alternate web sites, the first web site 1802 is defined to be accessed by clients having a high data transmission connection, e.g., a client having a T1 line. The second web site 1804 is defined to be access by clients having a low data transmission connection, e.g., a client having a 14.4 or a 28.8 kbps modem connection.

Figure 19 is an illustration of a web page sequence properties screen display 1900. The web page sequence properties screen display 1900 identifies the optimal data connection rate for a particular web site sequence. In Figure 19, the optimal user has a connection rate between 14.4 kbps and 33.3 kbps. The web page sequence properties screen display also enables the user to identify a slower alternate web page sequence and a faster alternate web page sequence that can be selected by the artificial intelligence unit, as described below.

Figure 20 is an illustration of a web page property screen display according to an embodiment of the present invention. The web page property screen display 2000 includes information about various components of each web page including the size and download time

of each component, whether the web page is static or dynamic, as defined above, and defines various timing features of the web page including the nominal load time, the ideal viewing time, the minimum viewing time and the maximum viewing time. In addition the page property enables the user to select various options including the display of control buttons, e.g., the forward and pause button, and enables the user to identify web pages that can be skipped if the actual data throughput is slower than the anticipated data throughput, as determined by the artificial intelligence unit, described below.

Figure 21 is an illustration of a sequence optimizer screen display according to an alternate embodiment of the present invention. The sequence optimizer screen display allows the user to optimize the arrangement of web pages within a web site and to optimize the length of time each web page is displayed. The sequence optimizer screen display is an easy-to-use graphical interface that displays the ideal and estimated viewing times for each page. The sequence optimizer is part of the web site configuration module 312 illustrated in Figure 3. The sequence optimizer of the present invention calls a web page display simulator that is also part of the web site configuration module 312 which provides feedback to the web site designer regarding the presentation of the sequence of web pages to the user under different network throughput conditions. The present invention determines the ideal and estimated viewing times based upon the amount of text in the web page, the level of detail in visual objects, and the length of video objects, for example. The user can modify the viewing time for each page.

Using this ideal viewing time and the estimated page loading time under various network throughput conditions, the Sequence Optimizer graphically displays for each page the amount of credit time, i.e., the time differential between the viewing time and the time at which the page has been preloaded, or the amount of debit time, the time differential between the time at which the page will be preloaded and the viewing time. The user can adjust the parameters for each page and see this credit/debit value change for each page. The parameters that can be set for each page include: the ideal view time, which is the amount of time the designer wants page to display, and the optional view flag, which indicates whether a page can be skipped over if the system is too busy, for example.

In addition the Virtual Design Studio unit automatically computes for each page the estimated minimum view time, which is the minimum time estimate that the page must be displayed before the next page is ready for display, and the normal load time, which is an estimate of the amount of time it will take to download a page in the throughput environment that the designer has set for this sequence.

The site layout studio unit is part of the web site configuration module 312 and enables the web site designer to create menus and define the web page sequence of the web site. The site layout studio unit displays a site menu page display and a menu item that displays the name of the site. When a user selects the menu item, one of at least two things happen: (1) a sequence of pages will play, or (2) another menu will appear, each of which can contain up to ten menu items. Using this model, sequences and menus can be nested infinitely. Figure 22 is an illustration of the site layout screen display that is displayed to the web site designer according to an alternate embodiment of the present invention. The site layout screen display 2200 includes a menu portion 2202, a menu layout portion 2204, and a control portion 2206. The menu portion 2202 includes the tree of menus and menu items within a site and the links there between. Selecting an object expands it and shows the menus and items linked hierarchically below it. The menu layout portion 2204 displays the screen layout of the menu items on each menu. Selecting a menu in the upper screen causes that menu to be displayed in the menu layout portion 2204. The user has the option of changing the menu style, by selecting the menu style button 2208 in the control portion 2206 in terms of the types of standard menu styles supported by the present invention, e.g., by modifying the displayed graphics and colors.

The virtual design studio unit includes a relational database that resides on the same computer the Virtual Design Studio is on, which can be the web site designer's computer. This database contains all the information regarding the automated web site except for the pages themselves, which are stored in files on the web server. A conventional HTML Generator module of the Virtual Design Studio creates an encoding of the database contents into a format that can be easily read by the client-side software on the user's computer. When the web site designer has completed defining the menus, pages, and sequences for the automated web site, the web site designer runs the HTML generator to create a file which is placed on the web server computer. These generated files contain the HTML code which the client-side processor and AI processing module uses to perform.

The web page preload module 1106 is described above with reference to Figures 11-13. In an alternate embodiment the web page preload module includes an artificial intelligence (AI) module that performs the functions described below. The artificial intelligence module supports the selection of different page versions and additional real-time decisions on page preloading. In other embodiments the artificial intelligence module can be stored on the server instead of the client computer.

After a user connects to the server having the web site, the AI module is downloaded to the client computer and collects information about the client computer environment including the operating system, processor capacity, Java support capabilities, types of video and audio formats that are supported, and the current throughput rate of the Internet connection. Using this information the AI module makes decisions regarding which version of Java to use and which version of the page sequence to use. Since network conditions can change during the session, the AI module continually monitors the network conditions, e.g., the rate of data throughput, and can modify the web page sequence by skipping a web page, for example, in order to reduce the waiting time for the user. The AI module periodically assesses the state of the session, and makes decisions accordingly. The functions performed by the AI module includes: (1) preloading pages; (2) canceling the preloading process; (3) modifying the rate of play of the current sequence; (4) inserting an additional page, e.g., a message, into the sequence; (5) skipping a page in a sequence being presented; (6) replaying one of the components, e.g., sound or video; and (7) switching to presenting another sequence of pages with a different ideal throughput requirement. A more detailed description of the AI module operation is described below.

In the alternate embodiment the AI module is downloaded to the client computer. Specifically, when the user contacts the web site, a client module is downloaded on the first page and the HTML code output by the HTML generator module of the virtual design studio, described above is also downloaded. This HTML code is used by the client code module and the AI module, and contains the complete description of the web site in terms of the menus, menu items, sequences, and pages found in each sequence. The client module performs three major functions: (1) probing the client machine to determine the client environment and the throughput rate, and uploading the HTML code; (2) displaying menus and pages; and (3) processing user toolbar input.

The client module implements the site template as designed by the web designer. The first menu page is displayed, and the user can choose one of the available options. When the user selects an option, the sequence of web pages associated with that option is displayed in sequence on a display device, e.g., a computer monitor, coupled to the client computer. Each web page can be displayed for a certain period of time, and the client module is responsible for switching pages after the view time has expired.

As stated above, some web pages may have an active toolbar. For example, some pages with a great deal of text may require that the user indicate when the user has finished reading the

text. It is possible to have a single view time that would work for all users, however fast readers may become frustrated if the view time is too long since the view time would need to be long enough to accommodate slower readers.

The client-server interaction and processing on the Internet World Wide Web is complex, and is greatly affected by factors such as network traffic, the number of clients accessing a particular site, the amount of content the client requested, the computer hardware speed, and the software efficiency. While some of these factors (e.g. modem speed) stay the same during an Internet session, other ones, such as network traffic, varies constantly.

To deliver an uninterrupted or nearly uninterrupted presentation to the user, the AI module frequently monitors the network, the server, and the client environment during a session, and makes complex decisions regarding how to proceed with the presentation. The AI module accounts for processes happening both on the server and on the client computer. The AI module synchronizes these processes while gathering information and while making decisions. The AI module constantly performs "what-if" scenarios, examining different options for preventing interruptions to the pages appearing on the user's display screen.

The AI module relies on the sequences being constructed in an optimal manner for nominal throughput for a particular environment. For example, if the nominal throughput for the sequence is 20 kbps, the AI module presumes that the sequence was constructed to flow smoothly as long as the connection uniformly maintains this throughput. However, the present invention is designed to work under various network conditions, under different server responsiveness, and with a variety of client computer environments, e.g., low-end personal computers with basic web browsers or high-end workstations supporting more powerful web browsers.

As described above, at the beginning of the client session, the AI module determines the client's environment and the current state of the Internet connection. Some of the parameters which will affect the decisions of the AI module are: (1) the processing and Input/Output (I/O) speed of the server machine; (2) the throughput capabilities of the Web Server software; (3) the operating system on the client machine; (4) the client computer's support of Java applets; (5) the client computer's support of JavaScript; (6) the client computer's support of Client Pull technology; (7) the types of video and audio formats that can be played by the browser or plug-in components of the client computer; (8) the maximum number of simultaneous connections to the Server that the client permits; (9) the browser cache size of the client computer; (10) the

number of users currently communicating with the Web Server; and (11) the current throughput rate of the Internet connection.

In one embodiment of the present invention, the AI module determines the following parameters at the beginning of a session and monitors the following parameters throughout the session: (1) the number of users currently communicating with the Web Server; (2) the current throughput rate of the Internet connection; and (3) the volatility of the network throughput rate.

At the beginning of each presentation, the AI module, based on the parameters of the session: (1) selects the versions of the requested presentation that are supported by the client environment, e.g., Java, JavaScript, ClientPull, or Generic; and (2) selects the starting version of the presentation. As described above, since network conditions may change during the presentation, the present invention can utilize all of the selected versions by, for example, switching between web page sequence versions in response to a variations in network conditions. In addition, the present invention sets the following parameters: (1) the maximum number of simultaneous connections the present invention utilizes; (2) the rate of change of the speed of the presentation sequence in response to change in network conditions; and (3) the types of filler pages, filler applications and special effects to be used during the presentation. In addition, as the session progresses, the AI module preloads new pages, graphics, sound, video and other presentation components. The AI module uses the information obtained during its network monitoring operation to determine whether the original sequence still can be played without interruptions. The AI module also monitors the progress of each component being preloaded from the Web Server, in order to make decisions about when to cancel preloading, or initiate new component preloading, as described above.

Appendix A includes a psuedo-code representation used to implement the AI module and is incorporated by reference herein.

The following is an example of the operation of the AI module. After the client connects to a server, the present invention determines, for example, that the client computer utilizes the Windows 95 Operating System, supports Java applets, supports running QuickTime videos, permits a maximum of two simultaneous connections to the server, and has a browser cache size of 10 megabytes. The AI module determines that versions of the top-level presentation can utilize Java applets and QuickTime video technology.

A probe page is downloaded to the client computer, and the present invention determines that the number of users currently communicating with the web server is at about half of Server's capacity and the current throughput rate of this Internet connection is 1000 kbps.

The present invention then selects the starting version of the presentation that corresponds to network throughput rate of 1000 kbps, and preloads the first page of the web site. The AI module determines that there are two components currently being downloaded, one from the next page, and one from the following page.

The presentation begins, and the present invention starts presenting selected sequence of pages, while continuing to monitor the presentation and network conditions.

During the presentation, and as a result of the network monitoring performed by the AI module, the present invention detects a drop in network connection's throughput, for example down to 100 kbps.

The AI module of the present invention sets up a virtual scenario of canceling the process of preloading the second component because even though the first component would download sooner, the timing for the next page would still be late.

The AI module determines that current page has an animated GIF component that can be replayed at the end of the page and adds this replay to the scenario being considered. However, invoking a what-if analysis still results in unacceptable timing. Therefore, the expert system consults a presentation script, and determines that at this point a "Skip page", or "Special Effect" actions are not available and, accordingly, rejects this scenario.

The expert system then runs through some other viable "what-if" scenarios, and rejects all of them except the one requiring switching to the sequence corresponding to a 56 kbps throughput.

The AI module then instructs the Vayu Web client module 1104 to modify the selected sequence of pages. The AI module then continues monitoring the presentation and network conditions.

Later in the presentation, the AI module detects an increase in network connection's throughput up to 900 kbps. The expert system sets up a virtual scenario of switching to the sequence corresponding to 1000 kbps throughput, and runs a what-if analysis on that scenario, whose result is acceptable. The AI module then switches back to the sequence of pages corresponding to 1000 kbps throughput, and continues with the presentation.

While the present invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by persons skilled in the relevant art that various changes in form and details can be made therein without departing from the spirit and scope of the invention.

Appendix A

The following is Java-based pseudo-code for the AI module for an embodiment of the present invention.

Class Definitions

```
class PrSite
{
      //describes the site (e.g. "General Goods")
      //for which presentations are built
      string                  name;
      PrSiteViewVec &     siteViewVec; //presentation projects
};


class PrSiteView
{
      //hierarchical tree of presentations accessible to user logging-in
      string                  name;
      PrSite &                site;
      PrPage &                rootMenuPage;       //top-level menu
      Presentation &          rootPresentation;   //top-level presentation
};


typedef Vector<Presentation> PresVec

class Presentation
{
      //describes a presentation corresponding to a menu item
      string            name;
      string            type;         //sequence, white-paper, order-form
      PrSiteView &      siteView;
      Presentation &          parent;             //parent in the pres. tree
      PresVec &         childVec;     //first child in the pres. tree
      SequenceVec &     sequenceVec;  //list of alternate page-sequences
      PrPage &          tailMenuPage; //menu to display after presentation
      PrPageVec &       tailPageVec;  //list of pages tacked-on (e.g. order-
                                      //form)
};

class PrSequence
{
      //sequence of pages making up a presentation
      //a presentation may have several alternative sequences
      //for different communication speed, Java support, plug-in support
      string            name;
      string            type;
      string            environmentType;//e.g. for Java-supporting browsers
      Presentation &          presentation; //back pointer to presentation owning
      PrSeqItemVec &          itemVec;      //list of items (pages) comprising this
};

class PrSequenceItem
{
      //item of the presentation sequence
      PrSequence &      sequence;
      PrPage &          page;         //page (usually HTML)
      long              idealViewTime;//prescribed under normal state
      long              minViewTime;
      long              maxViewTime;
      long              loadTime;     //expected load time at throughput
                                      //for which sequence was designed
```

```
        int             pauseAllowed; // user pauses when this item is displayed
        int             forwardAllowed;//enable "forward" button when displayed
        int             skipAllowed;  //allow skipping this page
        PrPage &        preloadPage;  //when displaying this item, start preloading
                                      //preloadPage before preloading next page
        PrFxVex      &  fxVec;              //list of available "special effects"
                                      //to be used when timing is Late
        PrSequence &    upSequence;   //another sequence to switch to
                                      //when Internet conditions improve
        PrSequenceItem & upItem;              //item within sequence to switch to
                                      //when Internet conditions improve
        PrSequence &    downSequence; //another sequence to switch to
                                      //when Internet conditions degrade
        PrSequenceItem & downItem;    //item within sequence to switch to
                                      //when Internet conditions degrade
};


class PrPage
{
        //HTML page, or special effect page, or applet
        string                  name;
        string                  type;  //HTML, VRML, Applet, FX
        string                  url;   //locator
        long                    size;  //file size (for download monitoring)
        PrComponentVec &    componentVec; //list of components (images, sound, video)
};

class PrComponent
{
        //image, video, sound, applet and other non-text components of a page
        string                  name;
        string                  mime_type;    //image, sound, etc.
        long                    size;         //file size
        long                    playTime;     //play (display) time; for timimg
        string                  url;          //locator
};

class PrFx
{
        //special effect (filler to smooth out the presentation at changing conditions)
        string                  name;
        string                  type;         //text msg, applet, animation clip
        string                  selectionType;      //random, by priority, specific
        PrPageVec &             pageVec;      //page containing this FX
};


class PrEnvironment
{
        //class of browser environment
        //for which sequences can be created
        string                  name;
};


#include "VauyWebPresSmooth.h"


class PresentationManager : public Applet
{
        int ProbePage()
        {
                //spawn a separate thread to gauge throughput rate
                spawnThread( "ThroughputTest");
```

```
        //get maximum number of simultaneous connections
        //the browser is allowed to open
        maxConnectionCnt = Browser.getMaxConnectionCnt();

        //get browser's buffer size
        netBufferSize = Browser.getNeetBufferSize();

        //inquire what kinds of plugins the browser has
        Browser.getPlugins(pluginVec);
        for( plugin = pluginVec.firstElement;
               plugin != NULL;
               plugin = plugin.nextElement())
        {
               switch plugin.type
               {
               case pluginStreamingVideo:
                      streamVideoType = StreamVideoTypeVec[plugin.vendor];
                      break;
               case pluginVirtualReality:
                      vrPlayer = Yes;
                      break;
               }      //switch
        }

        // wait for completion of the throughput test thread
        waitFor( finish, "ThroughputTest");


        //now we have all parameters needed to determine
        //which of the presentation environment to choose
        environ = determineEnvironment();

        //preload the top-level menu, and display it
        spawnThread( "PreloadPage", presView.rootMenuPage);
        curPage = presView.rootMenuPage;
        displayPage( curPage);

        //determine which sequences will potentially be used
        //in the presentation, as well as starting sequence
        rootPresentation.startSequence = DetermineSequences( sequenceVec);

        //preload first page of the chosen sequence
        spawnThread( "PreloadPage", "Fx", presView.rootPresentation);

        //spawn controlling thread
        spawnThread( "rootSequenceManager");

        //spawn a separate thread to monitor throughput rate
        spawnThread( "PeriodicThroughputTest");

}       //ProbePage()


int Presentation::SequenceManager()
{
        curSequence = startSequence;
        curEnv = startEnv;

        //go through all items of the sequence
        seqItem = mainSequence.pageVec.firstElement;
        while( true)
        {
               //reassess the state of the network (since prev. page)
```

```
curState = assessState();
switch curState
{
case stThroughputDidNotChangeDramatically:
      //can we preload one more component?
      if( preloadedCount >= MaxPreloadedCount)
      {
            break;
      }
      nextPreloadSeqItem =
            curSequence.getNextPreloadSeqItem();
      nextPreloadComp =
            curSequence.getNextPreloadComponent();
      //check if this comp is already cached
      nextPreloadComp.calcDownloadTime( curEnv);
      nextDownloadOK = true;

      //for each comp being downloaded,
      //see if starting new download will thwart timing
      for(i = 0; i < MaxPreloadedCount; ++i)
      {
            comp = componentBeingDownloaded[ i];
            if( comp.status != stInProgress)
                  continue;
            comp.calcTimeToStart();
            comp.calcRemainingTimeToDownload();

            if( comp.timeToStart <
                        comp.remainTimeDL +
                        nextPreloadComp.totTimeDL)
            {
                  //timing = Late; it is not OK to download
                  nextDownloadOK = false;
                  break;
            }
      }
      if( !nextDownloadOK)
            break; //switch
      //OK to download this component; do it
      spawnThread( nextPreloadComp.preload);
      break; //switch

case stUrlNotFound:        //file not found
      //can we skip this page?
      if( nextPreloadSeqItem.skipAllowed)
            //yes: skip it
            nextPreloadSeqItem.setSkip();
      else
            //no: try to switch sequences
            nextPreloadSeqItem.setJumpSequence();

case stThroughputIncreased:
      //keep count of how steady is the increase
      ThroughputIncreasedCount++;
      ThroughputDecreasedCount = 0;
      nextSequence.nextItem = curSequenceItem.upSeqItem();
      //can we afford (timing-wise) to go up the sequence
      nextSequence.nextItem.calcTiming();
      if(nextSequence.nextItem.timingOK)
            //yes: switch
            setNextItem(nextSequence.nextItem);
      else
            //no: act as if no change
            setStatus(stNormal);
```

```
                        break;

                case stThroughputDecreased:
                        //keep count of how steady is the decrease
                        ThroughputDecreasedCount++;
                        ThroughputIncreasedCount = 0;

                        calcCurTiming();
                        if(curSequenceItem.timingOK)
                                //we are still OK for this and next page
                                break;

                        //try different ways to handle oncoming delay
                        //(returns success if we can handle it)
                        if( (status = handleLateTiming()) == Success)
                                break;

                        //cannot handle delay with any other method
                        //try to switch to lower sequence
                        curSequence = curItem.lowSequence;
                        nextItem = curItem.getLowSeqItem();

                        break;

                case stThroughputLow:
                        //keep count of how steady is the decrease
                        ThroughputDecreasedCount++;
                        ThroughputIncreasedCount = 0;
                        calcCurTiming();
                        if(curSequenceItem.timingOK)
                                //we are still OK for this and next page
                                break;
                        else
                        {
                                setStatus(stThroughputDecreased);
                                continue;  //next iteration picks it up and handles
it
                        }

                }       //switch
        }       //while
}       //manager()

int handleLateTiming()
{
        //if we currently are displaying text,
        //inclrease viewing time by 20%
        //(whehter it helps or not)
        if(curItem.page.mimeType = "text")
        {
                increaseViewTime(curItem, percentIncreaseForText);
                calcCurTiming();
                if(curSequenceItem.timingOK)
                {
                        return Success;
                }
        }


        //can we replay any components already loaded?
        if( curItem.replayCompVec != NULL)
        {
                //rule: if comp i is replayed,
                //then i+1, i+2, ... must be replayed
                comp = curItem.replayCompVec.lastElement();
```

```
                while (comp != NULL)
                {
                        virtualAddComp( comp);
                        calcCurTiming();
                        if(curSequenceItem.timingOK)
                        {
                                curItem.setReplay(comp);
                                return Success;
                        } else
                                //try one more replay-components
                                comp = comp.PreviousElement();
                }
        }
        //are we preloading two pages ahead?
        //see if stopping that will help
        //if it does not help, do NOT stop preloading it yet!
        while(1)
        {
                seqItem = getLatestItemForWhichCompBeingPreloaded();
                if( seqItem != curItem
                        && seqItem != curItem.nextElement)
                {
                        virtualStopDownload(seqItem.downloadingComponents);
                        calcCurTiming();
                        if(curSequenceItem.timingOK)
                        {
                                //we will be OK for this and next page
                                stopDownload(seqItem.downloadingComponents);
                                return Success;
                        }
                } else
                        break;
        }       //while

        //can we skip next page (or couple)?
        while( curItem.nextElement.skipAllowed)
        {
                curItem.nextElement.setSkip();
                calcCurTiming();
                if(curSequenceItem.timingOK)
                {
                        //we will be OK for this and next page
                        return Success;
                }
        }
        //try FXs available
        if( curItem.FxVec != NULL)
        {
                fx = curItem.FxVec.firstElement();
                while( fx != NULL)
                {
                        virtualApplyFx( fx);
                        calcCurTiming();
                        if(curSequenceItem.timingOK)
                        {
                                //we will be OK for this and next page
                                return Success;
                        }
                }
        }
        return Fail;
}       //handleLateTiming()

}       //class
```

CLAIMS

What is claimed is:

1. A computer based method for generating a web site having a plurality of web pages, the computer having a storage module and having a wide area network interface, the method comprising the steps of:

designing a menu web page having one or more menu items;

designing a sequence of web pages, including a first and second web page, for each of said menu items, said first web page only capable of accessing one of said second web page and said menu web page, said second web page only capable of accessing the immediately subsequent web page in said sequence of web pages and said menu web page, said sequence of web pages;

identifying each menu item with one of said sequence of web pages; and

storing said menu web page and web page sequence information in the storage module.

2. The method of claim 1, wherein said sequence of web pages is designed by a webmaster, said sequence of web pages enables the webmaster to control the content and sequence of information displayed to the client.

3. The method of claim 2, further comprising the step of:

receiving a request signal from a destination computer via the wide area network interface, for requesting access to the web site;

transmitting said menu web page and said web page sequence information to said destination computer via the wide area network;

displaying said menu web page to a client on a display device coupled to said destination computer;

preloading additional web pages before said client selects a menu item;

receiving a menu item selection from a client;

preloading said sequence of web pages associated with said selected menu item, using a dynamic preload algorithm; and

displaying said sequence of web pages to the client without permitting the client to alter said sequence.

4. The method of claim 3, wherein said step of preloading said sequence of web pages includes the steps of:

determining a data receiving rate of the destination computer;

determining the size of two or more web pages in said sequence of web pages;

setting a priority value for each of said two or more web pages in said sequence of web pages based upon said data receiving rate, said size of said web page, and the position in said sequence of said web page;

preloading one of said two or more web pages into said destination computer storage module, based upon said priority value; and

dynamically modifying said priority value of one or more web pages to reduce any delay in displaying one of said web pages to the client.

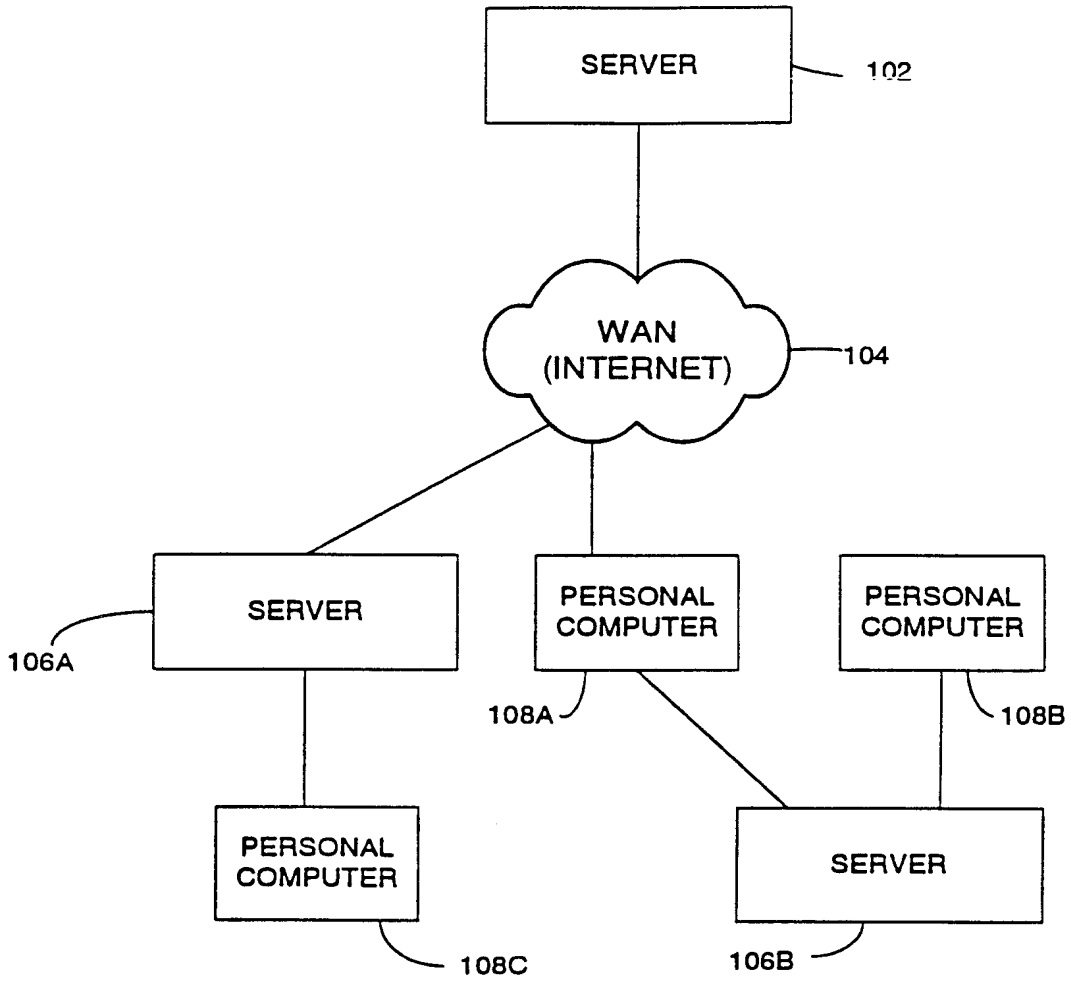

SERVER
102
WAN
(INTERNET)
104
SERVER
106A
PERSONAL
COMPUTER
108A
PERSONAL
COMPUTER
108B
PERSONAL
COMPUTER
108C
SERVER
106B

FIGURE 1

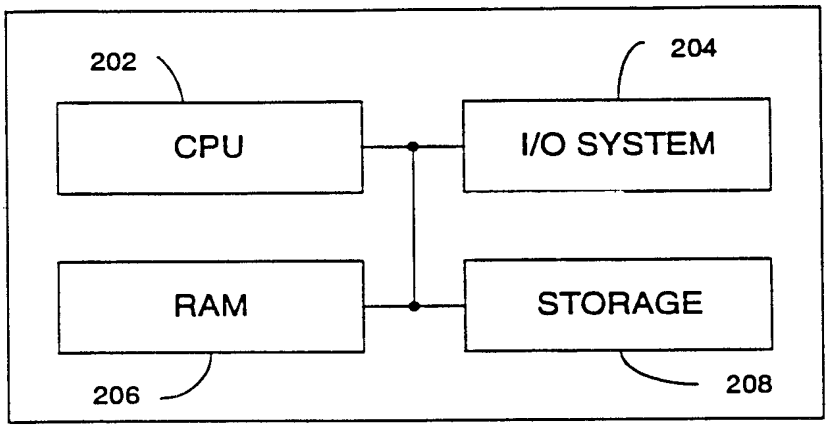


FIGURE 2

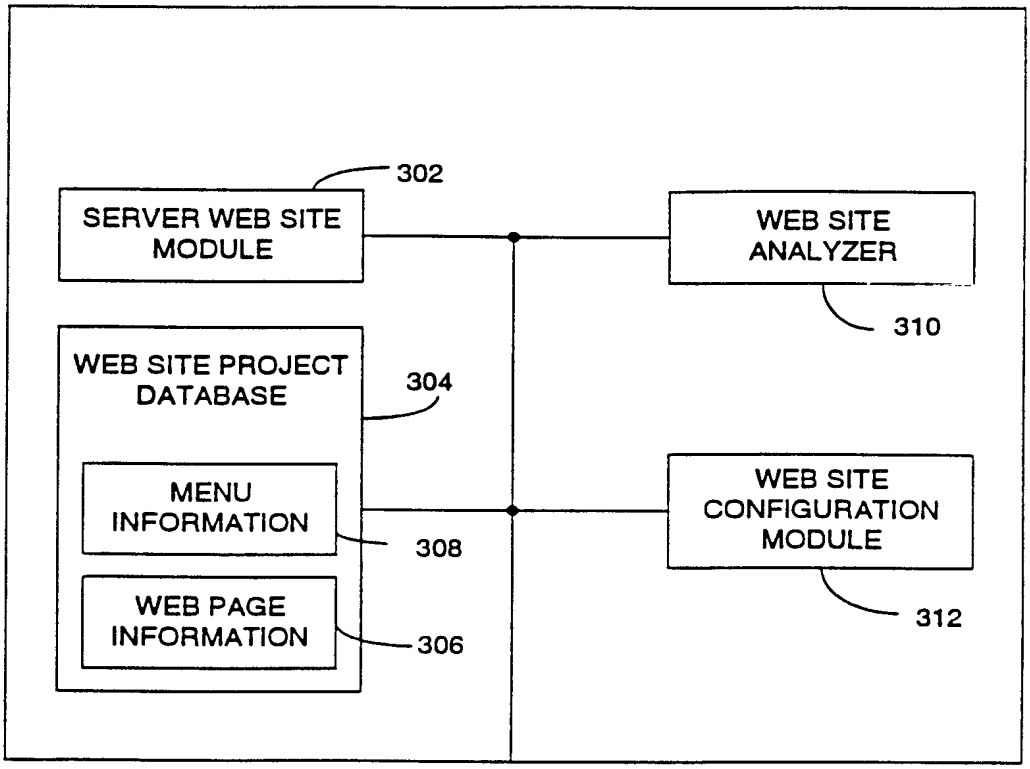

302
SERVER WEB SITE MODULE
WEB SITE ANALYZER
310
WEB SITE PROJECT DATABASE
304
MENU INFORMATION
308
WEB PAGE INFORMATION
306
WEB SITE CONFIGURATION MODULE
312

FIGURE 3

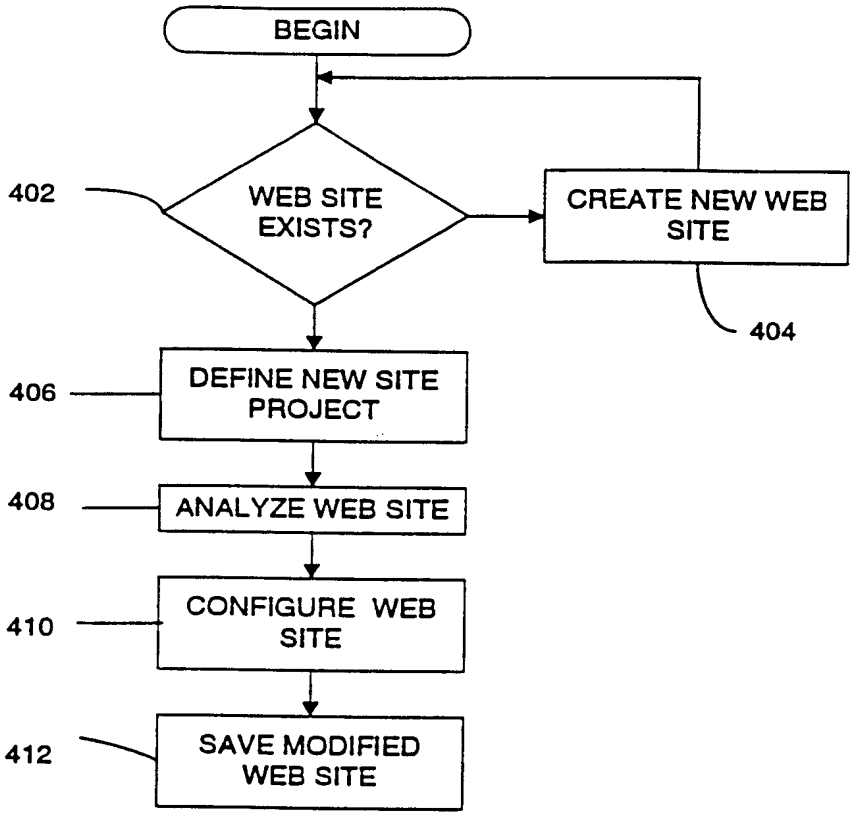

BEGIN
402
WEB SITE EXISTS?
CREATE NEW WEB SITE
404
406
DEFINE NEW SITE PROJECT
408
ANALYZE WEB SITE
410
CONFIGURE WEB SITE
412
SAVE MODIFIED WEB SITE

FIGURE 4

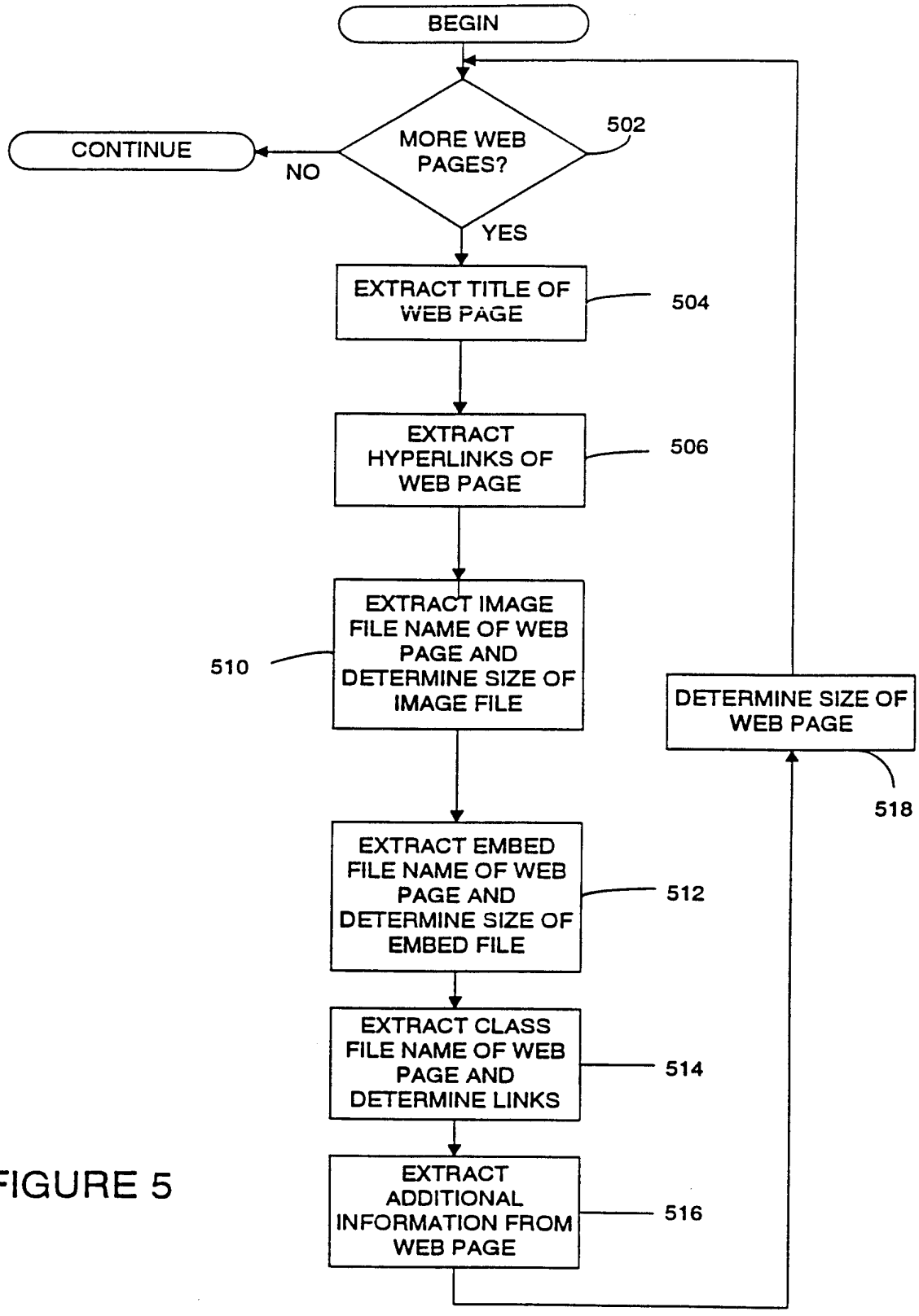

BEGIN
MORE WEB PAGES?
502
CONTINUE
NO
YES
EXTRACT TITLE OF WEB PAGE
504
EXTRACT HYPERLINKS OF WEB PAGE
506
EXTRACT IMAGE FILE NAME OF WEB PAGE AND DETERMINE SIZE OF IMAGE FILE
510
EXTRACT EMBED FILE NAME OF WEB PAGE AND DETERMINE SIZE OF EMBED FILE
512
EXTRACT CLASS FILE NAME OF WEB PAGE AND DETERMINE LINKS
514
EXTRACT ADDITIONAL INFORMATION FROM WEB PAGE
516
DETERMINE SIZE OF WEB PAGE
518

FIGURE 5

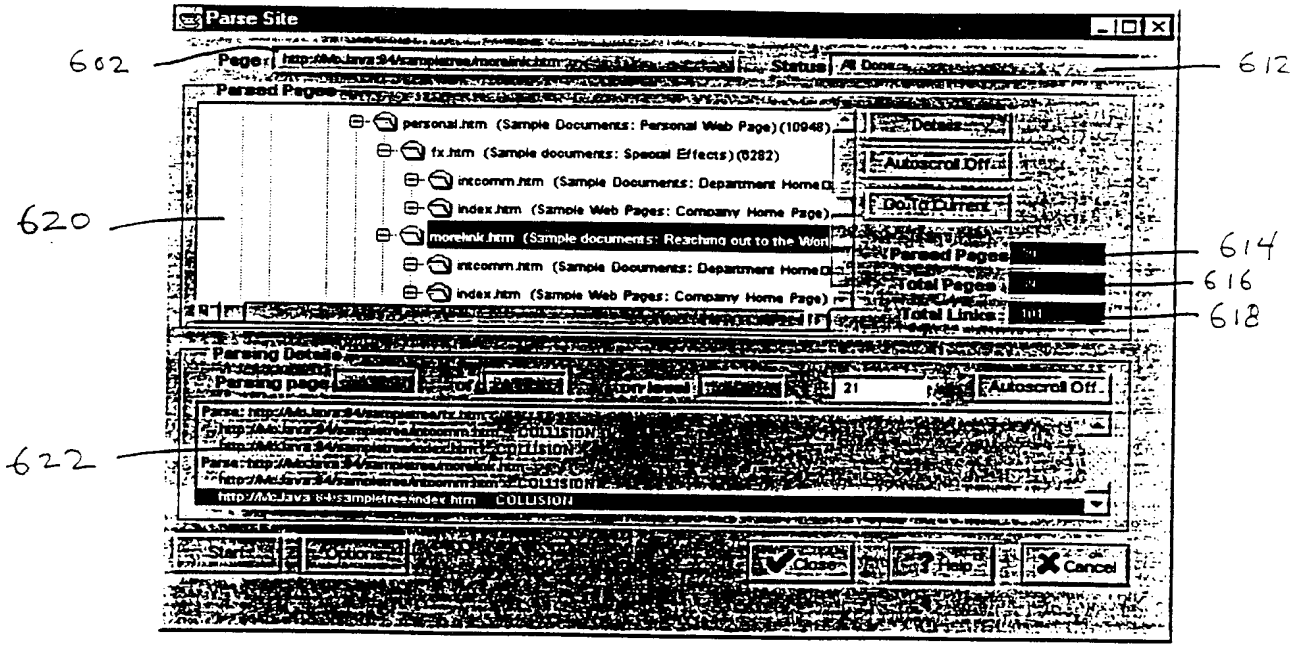

Parse Site
Pages
Status
All Done
Parsed Pages
personal.htm (Sample Documents: Personal Web Page) (10948)
fx.htm (Sample documents: Special Effects) (6282)
intcomm.htm (Sample Documents: Department Home
index.htm (Sample Web Pages: Company Home Page)
morelink.htm (Sample documents: Reaching out to the Wor
intcomm.htm (Sample Documents: Department Home
index.htm (Sample Web Pages: Company Home Page)
Details
Autoscroll Off
Do to Current
Parsed Pages
Total Pages
Total Links
Parsing Details
Parsing page
Autoscroll Off
COLLISION
Start
Options
Close
Help
Cancel
602
612
620
614
616
618
622

Figure 6

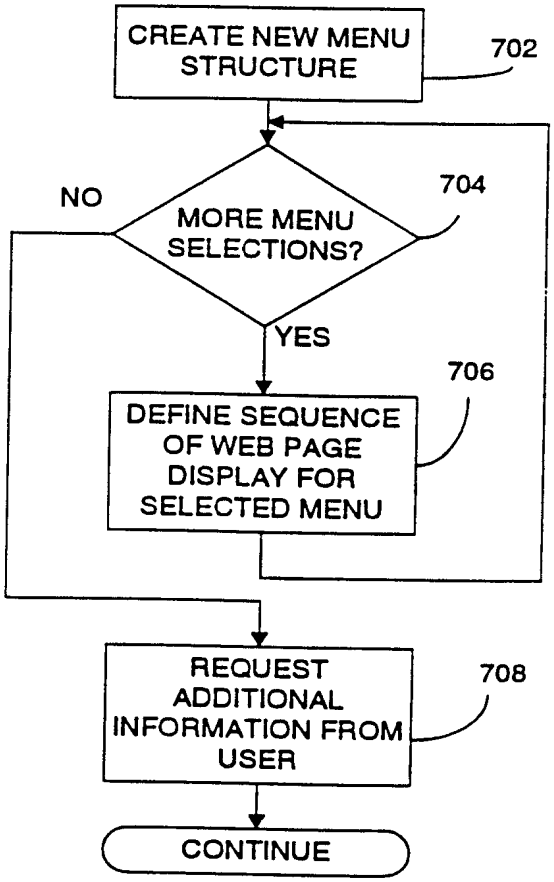

CREATE NEW MENU STRUCTURE
702
MORE MENU SELECTIONS?
704
NO
YES
DEFINE SEQUENCE OF WEB PAGE DISPLAY FOR SELECTED MENU
706
REQUEST ADDITIONAL INFORMATION FROM USER
708
CONTINUE
410

FIGURE 7

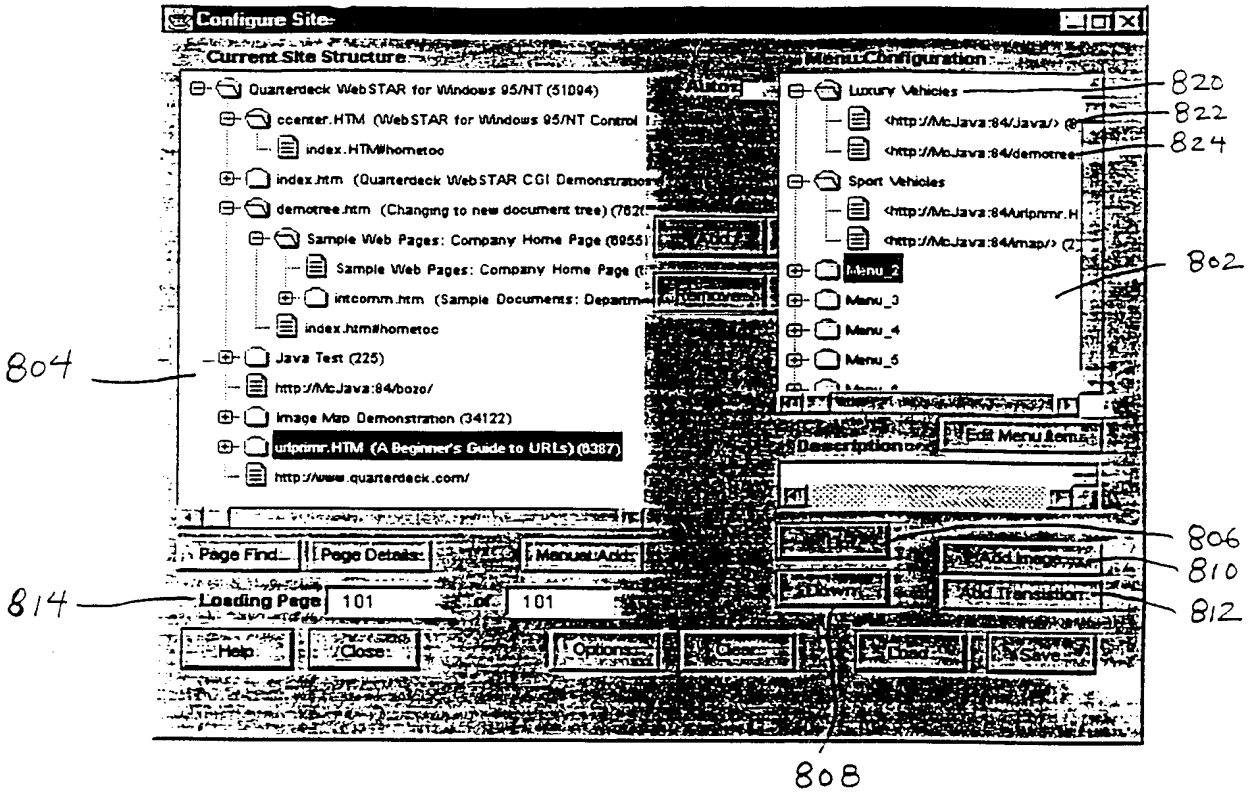

Configure Site
Current Site Structure
Quarterdeck WebSTAR for Windows 95/NT (51094)
ccenter.HTM (WebSTAR for Windows 95/NT Control
index.HTM#hometoc
index.htm (Quarterdeck WebSTAR CGI Demonstration
demotree.htm (Changing to new document tree) (762
Sample Web Pages: Company Home Page (8955)
Sample Web Pages: Company Home Page (
intcomm.htm (Sample Documents: Departm
index.htm#hometoc
Java Test (225)
http://McJava:84/bozo/
Image Map Demonstration (34122)
urlprimr.HTM (A Beginner's Guide to URLs) (8387)
http://www.quarterdeck.com/
Page Find...
Page Details:
Manual Add:
Loading Page 101 of 101
Help
Close
Options
Menu Configuration
Luxury Vehicles
<http://McJava:84/Java/> (8
<http://McJava:84/demotree
Sport Vehicles
<http://McJava:84/urlprimr.H
<http://McJava:84/imap/> (2
Menu_2
Menu_3
Menu_4
Menu_5
Description
Edit Menu Items
Add Image
Add Transition
Load
Save
820
822
824
802
804
806
810
812
814
808

Figure 8

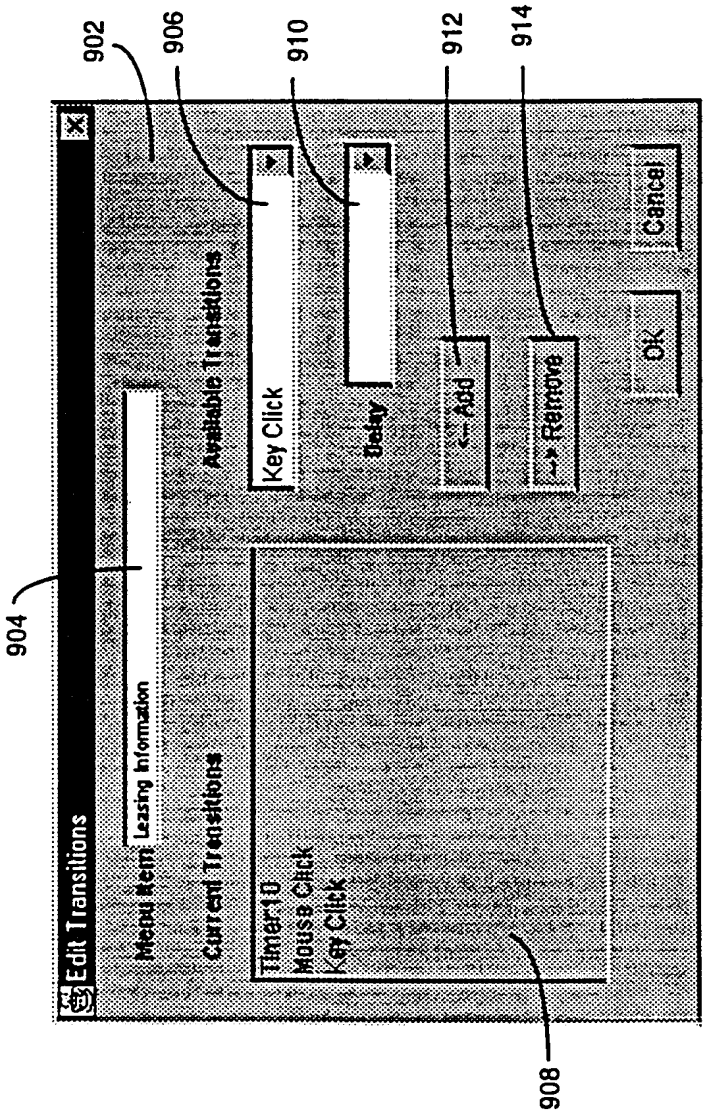

902
906
910
912
914
904
908
Edit Transitions
Menu Item
Leasing Information
Current Transitions
Timer:10
Mouse Click
Key Click
Available Transitions
Key Click
Delay
<-- Add
--> Remove
OK
Cancel

Figure 9

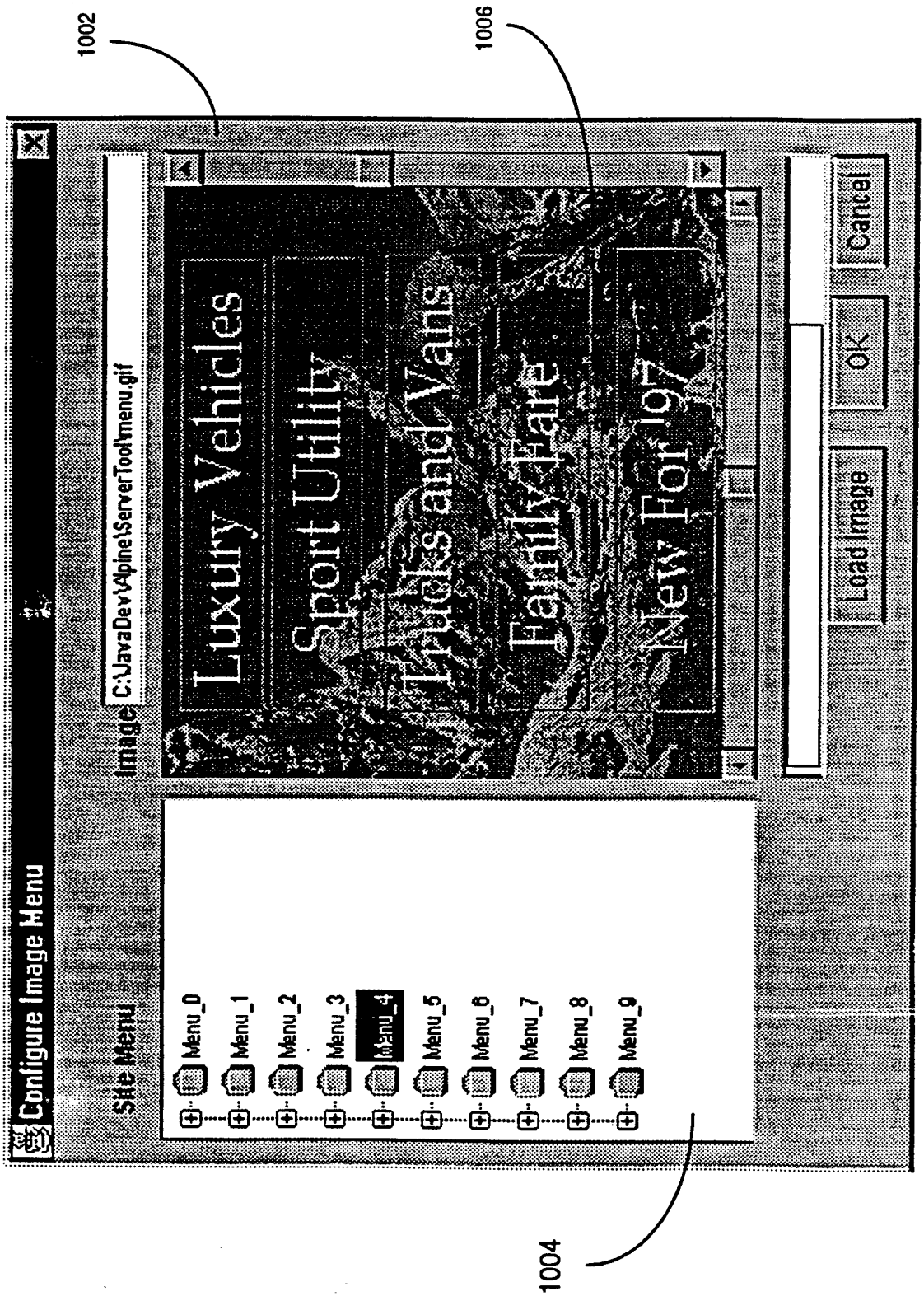

1002
1006
1004
Configure Image Menu
Site Menu
Menu_0
Menu_1
Menu_2
Menu_3
Menu_4
Menu_5
Menu_6
Menu_7
Menu_8
Menu_9
Image C:\JavaDev\Alpine\ServerTool\menu.gif
Luxury Vehicles
Sport Utility
Trucks and Vans
Family Fare
New For 97
Load Image
OK
Cancel

Figure 10

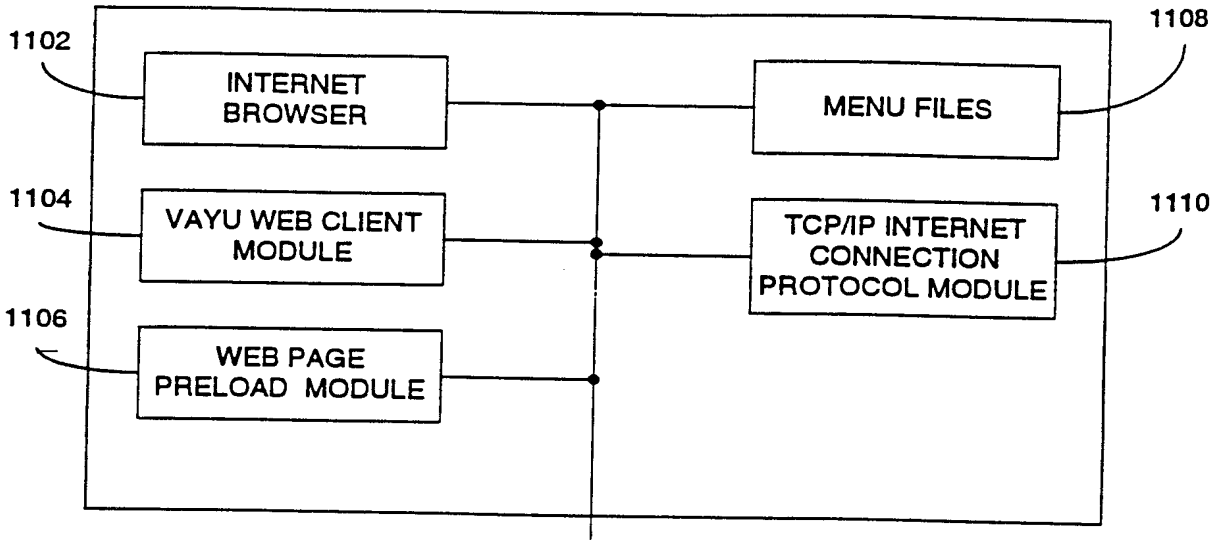

1102
INTERNET BROWSER
1104
VAYU WEB CLIENT MODULE
1106
WEB PAGE PRELOAD MODULE
1108
MENU FILES
1110
TCP/IP INTERNET CONNECTION PROTOCOL MODULE

FIGURE 11

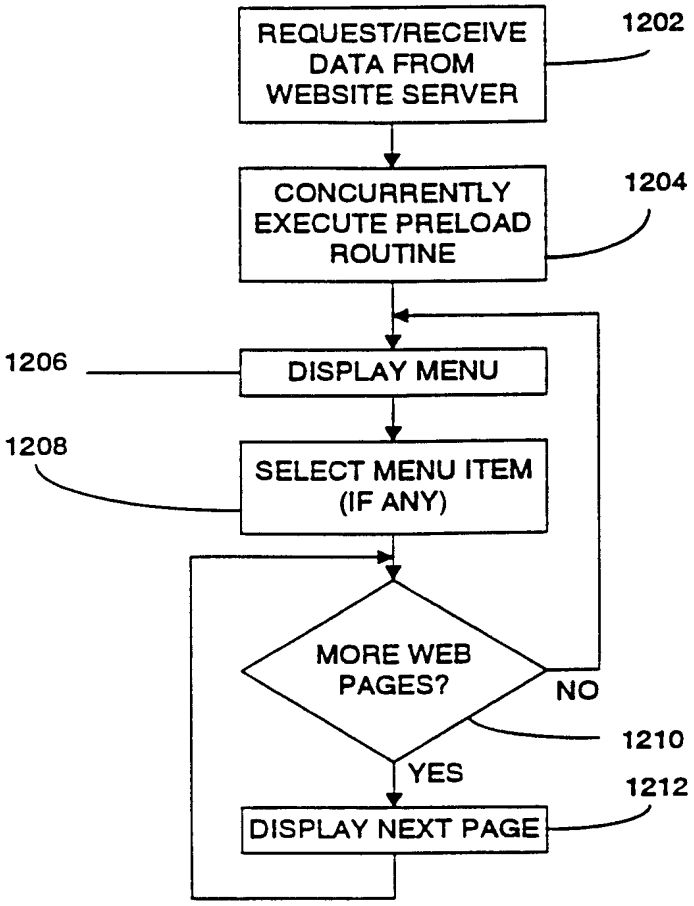

REQUEST/RECEIVE DATA FROM WEBSITE SERVER
1202
CONCURRENTLY EXECUTE PRELOAD ROUTINE
1204
1206
DISPLAY MENU
1208
SELECT MENU ITEM (IF ANY)
MORE WEB PAGES?
NO
1210
YES
1212
DISPLAY NEXT PAGE

FIGURE 12

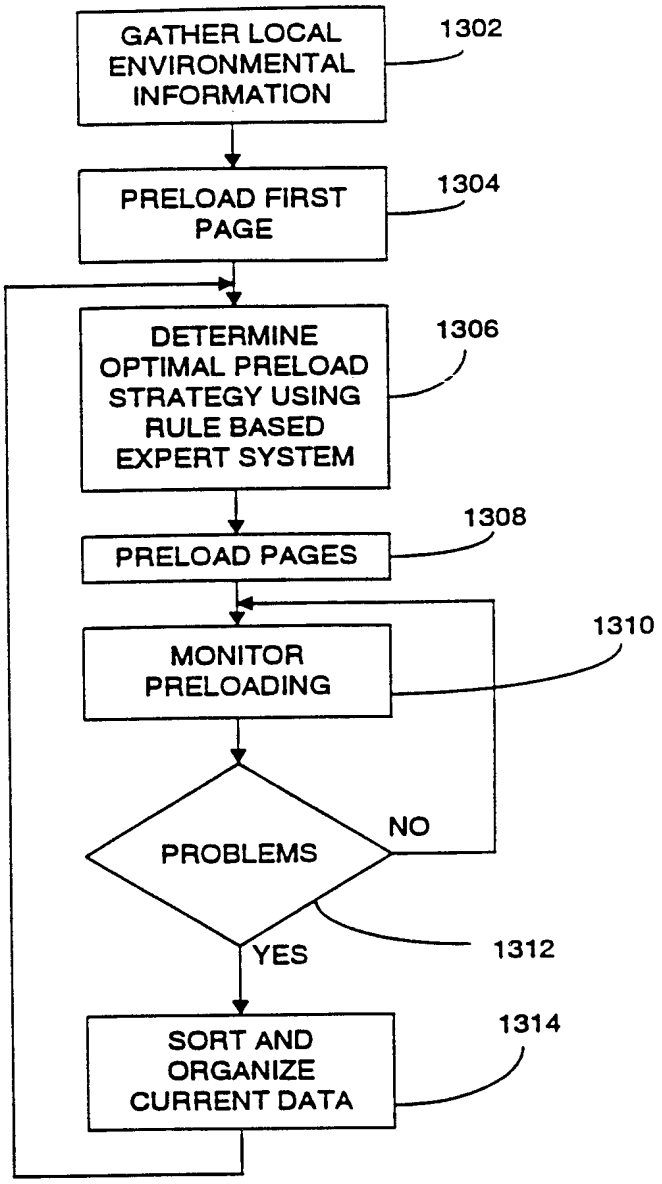

GATHER LOCAL ENVIRONMENTAL INFORMATION
1302
PRELOAD FIRST PAGE
1304
DETERMINE OPTIMAL PRELOAD STRATEGY USING RULE BASED EXPERT SYSTEM
1306
PRELOAD PAGES
1308
MONITOR PRELOADING
1310
PROBLEMS
NO
YES
1312
SORT AND ORGANIZE CURRENT DATA
1314

FIGURE 13

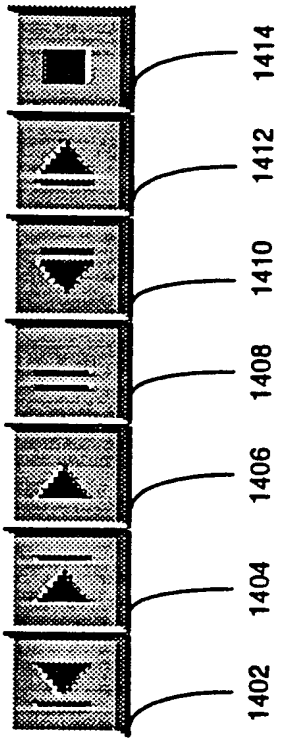


Figure 14

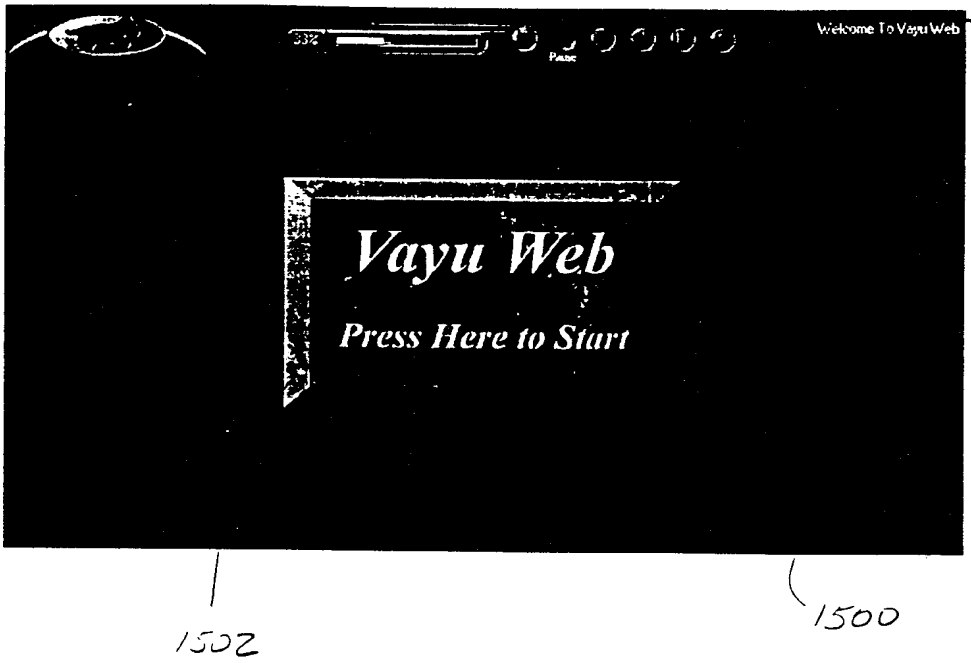

1504
Welcome To Vayu Web
Vayu Web
Press Here to Start
1502
1500

FIGURE 15

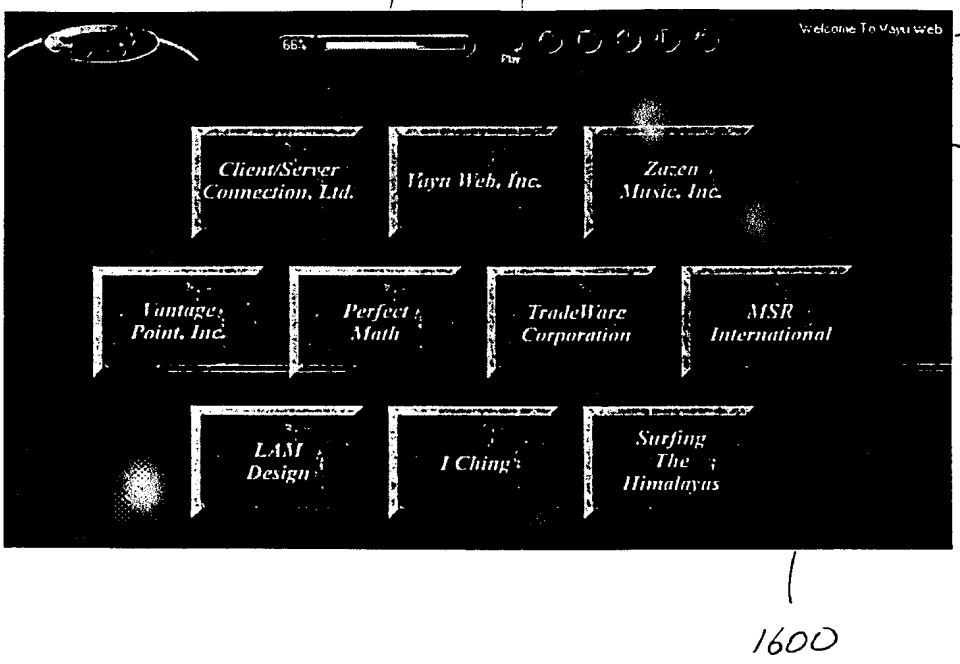

1606
1604
Welcome To Vayu Web
1504
1602
Client/Server Connection, Ltd.
Vayu Web, Inc.
Zazen Music, Inc.
Vantage Point, Inc.
Perfect Math
TradeWare Corporation
MSR International
LAM Design
I Ching
Surfing The Himalayas
1600

FIGURE 16

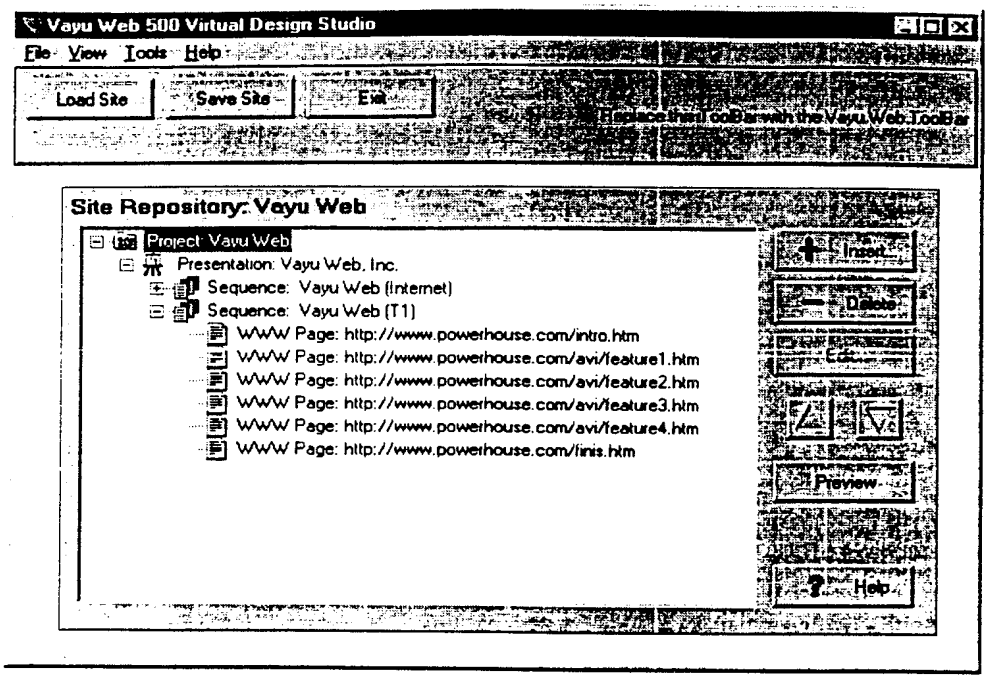


FIGURE 17

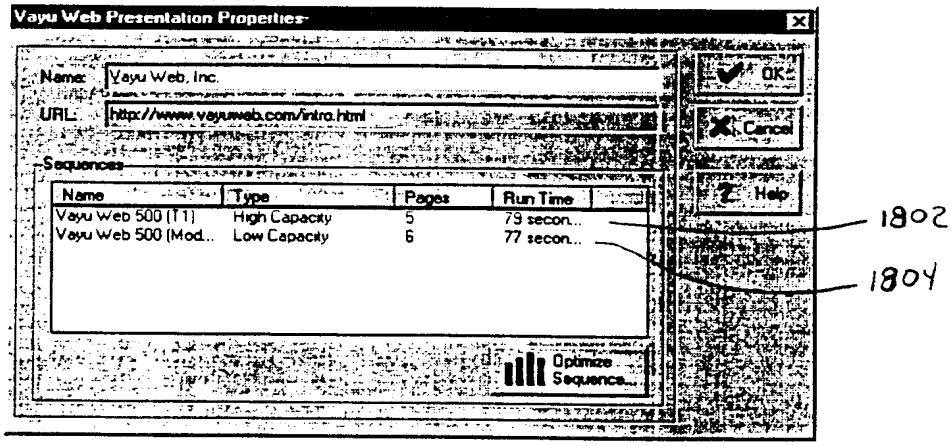


FIGURE 18

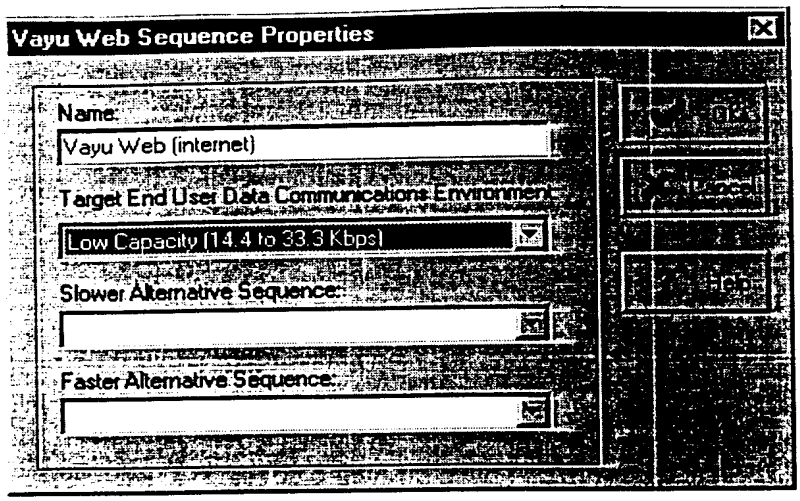


FIGURE 19

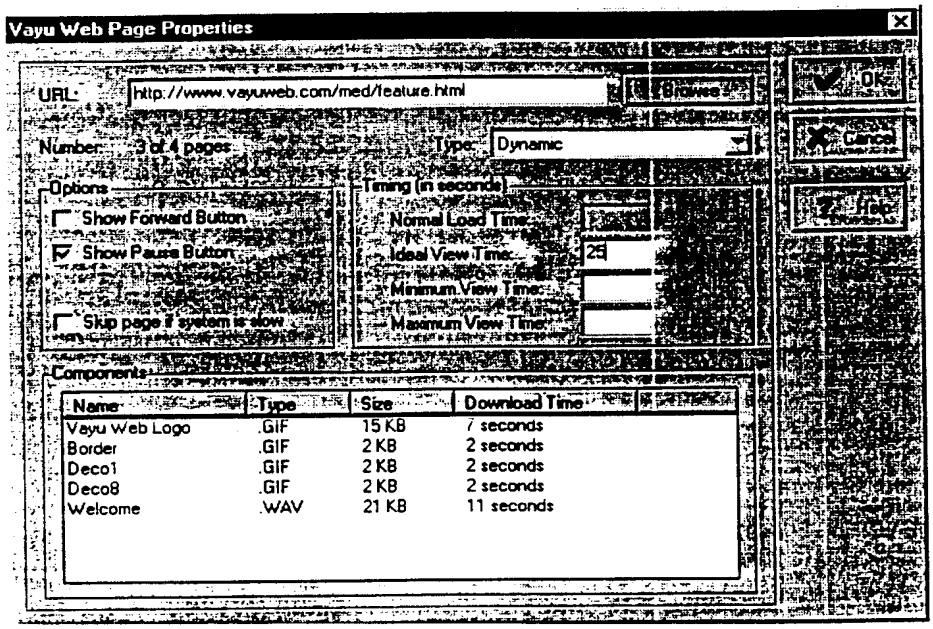


FIGURE 20

FIGURE 21

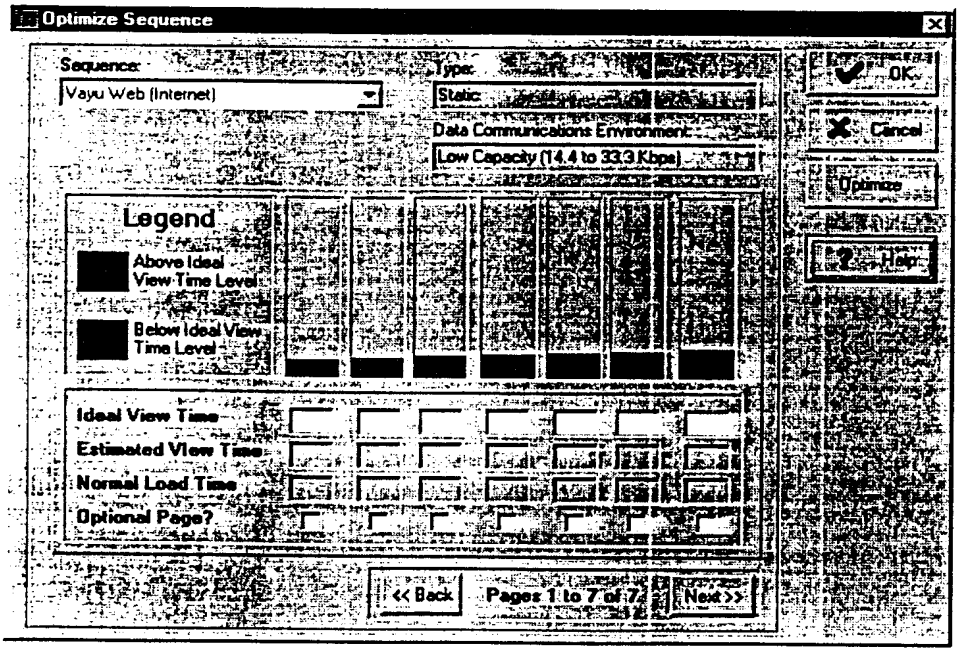
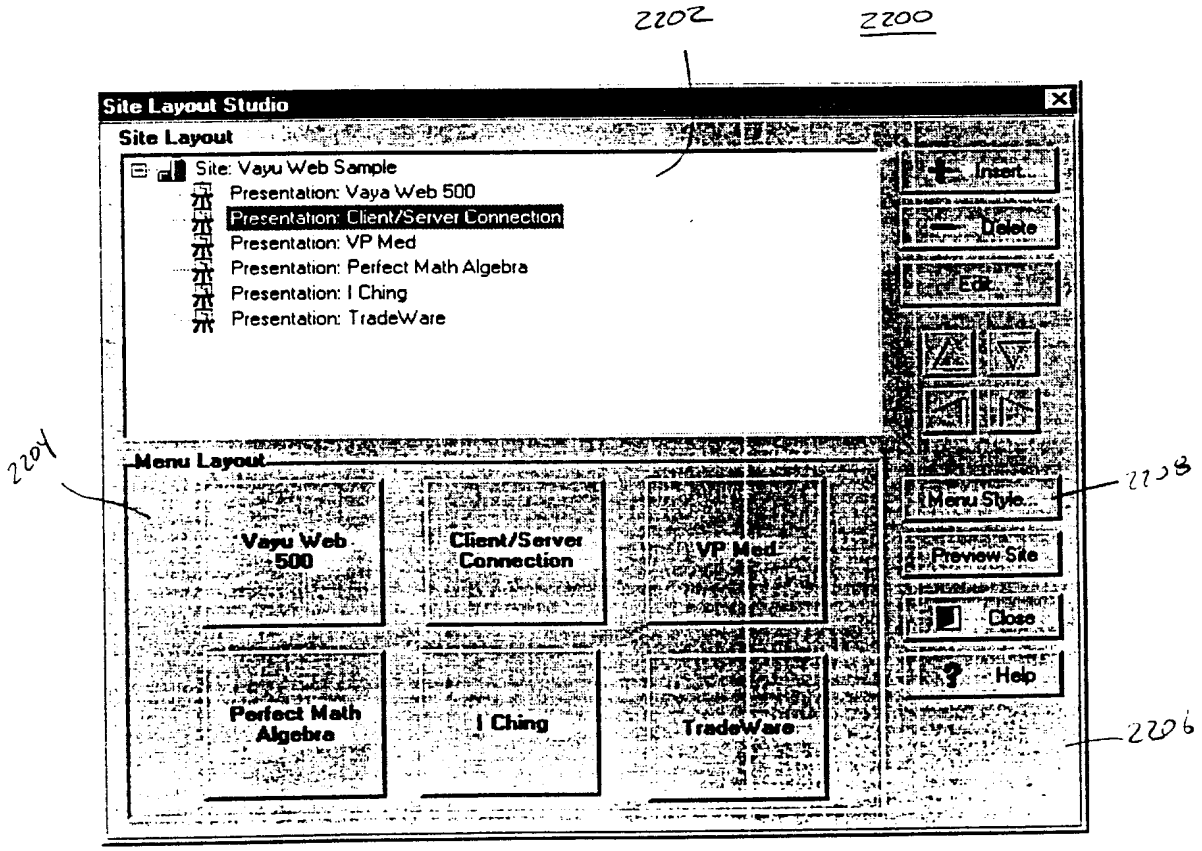

Optimize Sequence
Sequence:
Vayu Web (Internet)
Type:
Static
Data Communications Environment:
Low Capacity (14.4 to 33.3 Kbps)
OK
Cancel
Optimize
Help
Legend
Above Ideal View Time Level
Below Ideal View Time Level
Ideal View Time
Estimated View Time
Normal Load Time
Optional Page?
<< Back
Pages 1 to 7 of 7
Next >>

FIGURE 22


2202
2200
Site Layout Studio
Site Layout
Site: Vayu Web Sample
Presentation: Vaya Web 500
Presentation: Client/Server Connection
Presentation: VP Med
Presentation: Perfect Math Algebra
Presentation: I Ching
Presentation: TradeWare
Insert...
Delete
Edit...
Menu Style...
Preview Site
Close
Help
Menu Layout
Vayu Web 500
Client/Server Connection
VP Med
Perfect Math Algebra
I Ching
TradeWare
2204
2208
2206